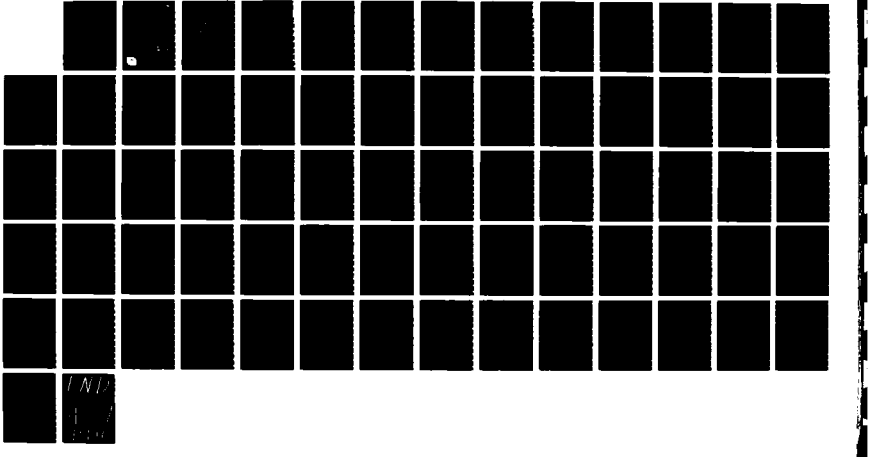
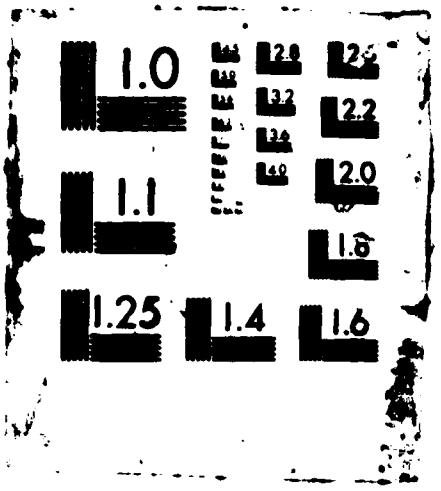
AD-A178 440 ADA (TRADE NAME) FOUNDATION TECHNOLOGY VOLUME 9 SOFTWARE REQUIREMENTS FOR..(U) INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA A AGRAWALA ET AL. DEC 86 1/1

UNCLASSIFIED IDA-P-1893-VOL-9 IDA/HQ-86-30825 F/G 17/2 NL




1.0
1.1
1.25
1.4
1.6
2.0
2.2

AD-A178 440

(2)

IDA PAPER P-1893

Ada* FOUNDATION TECHNOLOGY

Volume IX: Software Requirements for WIS Network Protocol Prototypes

Ashok Agrawala, *Chairman*
Michael Bloom, *IDA Task Force Manager*
Robert Boorstyn
R. J. Buhr
Deborah Heystek
Don Towsley
John Salasin, *Program Manager*

With Additional Contributions by
Dennis MacKinnon

December 1986

Prepared for
Office of the Under Secretary of Defense for Research and Engineering

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311

*Ada is a registered trademark of the U.S. Government, Ada Joint Program Office.

IDA Log No. HQ 86-38825

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA Paper does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and sound analytical methodology and that the conclusions stem from the methodology.

Approved for public release, distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE AD-A178440

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) P-1893 - Volume IX	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses 6b OFFICE SYMBOL IDA	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	7b ADDRESS (City, State, and Zip Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION WIS Joint Program Management Office 8b OFFICE SYMBOL (If applicable) WIS/JPMO	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c ADDRESS (City, State, and Zip Code) 7798 Old Springfield Road McLean, VA 22102	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-W5-206 / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification)
Ada Foundation Technology: Volume IX - Software Requirements for WIS Network Protocol Prototypes

12 PERSONAL AUTHOR(S)
A. Agrawala, M. Bloom, R. Boorstyn, R.J. Buhr, D. Heystek, D. Towsley, D. MacKinnon

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM _____ TO _____	1986 December	72

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
World Wide Military Command and Control System (WWMCCS), WWMCCS Information System (WIS), automatic data processing (ADP), local area network (LAN), command, control and communication (C3), Ada programming language, Open System Interconnection (OSI), Wide Area Network (WAN)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This document is the result of the identification of the functionality requirements and research of the technology base. Three specifications for prototypes in the foundation area of network protocols have been generated and support the stated objective of developing software in Ada to support WIS communications functionality in the 1990's.

The three specifications are: a) Common Ada Implementation of the OSI and DoD Transport and Internet Protocols (Section 2.0); b) Towards Automatic Generation of Ada Protocol Software for WIS (Section 3.0); c) Development and Evaluation of Multivariable Objective Function Network Routing for WIS (Section 4.0).

This volume is the last of a nine-volume set describing projects which are planned for prototype foundation technologies for WIS using the Ada programming language. The other volumes include command language; software design, description, and analysis tools; text processing; database management system; operating systems; planning and optimization tools; and graphics.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
■ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

IDA PAPER P-1893

Ada™ FOUNDATION TECHNOLOGY

Volume IX: Software Requirements for WIS Network Protocol Prototypes

Ashok Agrawala, *Chairman*
Michael Bloom, *IDA Task Force Manager*
Robert Boorstyn
R. J. Buhr
Deborah Heystek
Don Towsley
John Salasin, *Program Manager*

With Additional Contributions by
Dennis MacKinnon

December 1986

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-W5-206

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 Purpose

Several projects are planned to prototype foundation technologies for the World Wide Military Command and Control System (WWMCCS) Information System (WIS) using the Ada programming language. The purpose for developing these prototypes is to produce software components that:

a. Provide the functionality required by WIS.

b. Use the Ada programming language to provide the maximum portability, reliability, and maintainability consistent with efficient operation.

c. Are consistent with current and "in-process" software standards.

Foundation areas in which prototypes will be developed include:

a. Command Language

b. Software Design, Description, and Analysis Tools

c. Text Processing

d. Database Management System

e. Operating System

f. Planning and Optimization Tools

g. Graphics

h. Network Protocols

1.2 Scope

This document is the result of the identification of the functionality requirements and research of the technology base. Three specifications for prototypes in the foundation area of network protocols have been generated and support the stated objective of developing software in Ada to support WIS communications functionality in the 1990's.

The three specifications are:

a. Common Ada Implementation of the OSI and DoD Transport and Internet Protocols (Section 2.0)

b. Towards Automatic Generation of Ada Protocol Software for WIS (Section 3.0)

c. Development and Evaluation of Multivariable Objective Function Network Routing for WIS (Section 4.0)

Section 2.0, Common Ada Implementation of the OSI and DoD Transport and Internet Protocols, provides a technical definition of a project which will result in the production of a common implementation of the Open Systems Interconnection (OSI) and DoD transport and internet protocols. The objectives of this project are to provide DoD with information regarding the adoption of International Standards Organization (ISO) protocols, reduce both the risk and cost associated with this transition, and generally enhance the capabilities of DoD networks.

Two projects are described in Section 3.0, Towards Automatic Protocol Software Generation. These projects will establish a baseline for later developing a prototype system capable of taking protocol and program design specifications as input and producing programs for WIS protocols. The two projects, which may be pursued in parallel, involve gaining experience generating Ada code fragments from protocol specifications and comparing alternative approaches to solving the combined environment problem.

Section 4.0, Development and Evaluation of Multivariable Objective Function Network Routing, describes a project to develop prototype mathematical models which find routes that optimize multiple objectives subject to given constraints. The objective is to develop and evaluate, with respect to performance and efficiency, prototype routing algorithms which can be implemented and integrated into WIS.

Although the responsibility for the protocols has not yet been established as being with WIS or DoD, the development of these specifications for prototypes establishes a baseline for full scale development of the protocol software. The anticipated impact is a reduction in the cost of development and implementation of protocols for WIS and DoD. A further benefit will result from wide-area network functionality tailored to multiple application requirements.

1.3 Terms and Abbreviations

ANNA	Annotated Ada
AI	Artificial Intelligence
CCITT	Consultative Committee for International Telephone & Telegraph
CLNP	Connectionless Internet Protocol
DDN	Defense Data Network
DoD	Department of Defense
Estelle	Extended State Machine Language
ICMP	Internet Control Message Protocol
IDP	Initial Domain Part
IP	Internet Protocol
ISO	International Standards Organization
LAN	Local Area Network
LG	Local Gateway
NBS	National Bureau of Standards
OS	Operating System
OSI	Open System Interconnection
PDU	Protocol Data Unit
SNAP	Subnetwork Access Protocol
SNDCP	Subnetwork Dependent Convergence Protocol
SNICP	Subnetwork Independent Convergence Protocol
TCP	Transmission Control Protocol
VHSIC	Very High Speed Integrated Circuits
WAN	Wide Area Network
WIS	WWMCCS Information System

WIS OS WIS Operating System
WWMCCS World Wide Military Command and Control System

1.4 References

[AGRA 85] Agrawala, Ashok, personal communication with Ashok Agrawala of the U. of Maryland on a project to specify TCP/IP formally.

[ALFO 77] Alford, M.W., "A Requirements Engineering Methodology for Real-Time Processing Environments," *IEEE Transactions on Software Engineering*, SE-3, 1, January 1977.

[ALFO 84] Alford, M.W., "Distributed Computing Design System," presented at the Santa Barbara Workshop on Future Ada Environments, September 1984.

[BALZ] Balzer is understood to have developed at ISI a Temporal Behaviour Specification Language and associated code generator which is undergoing some commercial development by TRW.

[BALZ 83] Balzer, R., T.E. Cheatham, and C. Green, "Software Technology in the 1990's: Using a New Paradigm," *IEEE Computer*, 16, 11, (November 1983): 39-45.

[BERT 80] Bertsekas, D. P., "A Class of Optimal Routing Algorithms for Communication Networks," *International Conference on Circuits and Computers*, Atlanta, November 1980.

[BOCH 84] Bochmann, G.v., "Comparison of DoD and ISO/CCITT Transport Layers," Part II of *Report for Canadian Department of Communications*, Contract OST83-00311, November 1984.

[BOCH 85] Bochmann, G.v.,(1985), Estelle/Pascal translator - verbal communication.

[BOND] Bondeli, Patrick de, "Models for the Control of Concurrency in Ada Based on Predicate Transition Nets," *C.R.2.A.*, 18 rue d'Arras, 92000 Nanterre, France, undated report.

[BUHR 84] Buhr, R.J.A., *System Design With Ada*, Prentice Hall, 1984.

[BUHR 85A] Buhr, R.J.A., G.M. Karam, C.M. Woodside, "An Overview and Example of Application of CAEDE: A New Design Environment for Ada," *1985 International Ada Conference*, Paris, May 85.

[BUHR 85B] Buhr, R.J.A., C.M. Woodside, G.M. Karam, K. Van der Loo, G. Lewis, "Experiments With Prolog Design Descriptions and Tools in CAEDE: an Iconic Design Environment for Multitasking, Embedded Systems," *Eighth International Conference on Software Engineering*, London, August, 1985.

[BURC 83] Burchfiel, J., J. Westcott, and G. Lauer, "Distributed Routing Techniques for Closely Coupled Networks," BBN, October 1983.

[CADO 85] Verbal information about CADOS project from a visitor to British Telecom.

[CANT 74] Cantor, D.G., and M. Gerla, "Optimal Routing in a Packet Switched Computer Network," *IEEE Transactions on Computers*, October 1974.

[CAVA 84] Cavalli, A.R., "A Method of Automatic Proof for the Specification and Verification of Protocols," *Proceedings of the SIGCOMM 84*, Montreal, June 1984.

[CCITT 84A] *CCITT SDL Newsletter*, No. 6, January 1984, Special Issue on Languages for Protocol Specification, covering SDL, Estelle, Numerical Petri Nets and Lotos. Edited by R. Saracco, CSELT, v.Reiss Romoli 274, 10147, Turin, Italy. Distributed by Telecom Australia.

[CCITT 84B] *CCITT SDL Newsletter*, No. 7, Aug 1984, containing the Z.100 to Z.104 series of specifications on SDL. Edited by F. Belina, Telelogic AB, Baltzarsgatan 22, S-211 36 MALMO, Sweden. Distributed by the Swedish Telecommunication Administration.

[CHER 84] Cherry, George W., "Parallel Programming in ANSI Standard Ada," Reston, 1984.

[CLAR 84] Clarke, L. et al, paper on design environments at the Ada Minneapolis Conference, Fall 1984.

[COUR 81] Courtois, P.J., and P. Semal, "An Algorithm for the Optimization of Nonbifurcated Flows in Computer Communication Networks," *Performance Evaluation* 1 (1981): 139-152.

[DoD IP] U.S. Department of Defense, "Internet Protocol," MIL-STD 1777, 12 August 1983.

[DoD TCP] U.S. Department of Defense, "Transport Control Protocol," MIL-STD 1778, 12 August 1983.

[ELLI 83] Ellis, J.T., "ADL/ADS - A Testbed Tool for Experimentation With Real Time DDP Architectures", IEEE, 1983.

[FIDG 84] Fidge, C.J., G.J. Cain, L.N. Jackson, R.S.V. Pascoe, "An Overview of the MELBA Automatic Code Generation Project," *A.T.R.* 18, 1, 1984 (Australia).

[FRAT 73] Fratta, L., M. Gerla, and L. Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Network Design," *Networks*, 1973: 97-133.

[FREN 85] Frenkel, K.A., "Toward Automating the Software Development Cycle," *Communications of the ACM*, 28, 6, June 1985.

[GAFN 81] Gafni, E.M., and D. P. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes with Frequently Changing Topology," *IEEE Transactions on Communications*, January 1981.

[GALL 77] Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications*, January 1977.

[GAVI 83] Gavish, B., and S. L. Hantler, "An Algorithm for Optimal Route Selection in SNA Networks," *IEEE Transactions on Communications*, October 1983.

[HAIL 82] Hailpern, B.T., *Verifying Concurrent Processes Using Temporal Logic*, Lecture Notes on Computer Science, No. 129, Springer- Verlag, 1982.

[HANT 85] Hantler, S.L., H.B. Putz, B.A. Taylor, "Lower Bounds for High Priority Delay in Telecommunication Networks," IBM Research Report, RC 10959, January 1985.

[ISO CLIP] International Standards Organization, *Protocol for Providing the Connectionless Network Service*, ISO DIS 8473.

[ISO CTM] International Standards Organization, *Working Draft for OSI Conformance Testing Methodology and Framework*, ISO TC97/SC21 N410.

[ISO NA1] International Standards Organization, *Information Processing Systems - Data Communications - Network Service Definition - Addendum 1*, ISO 8348/DAD1.

[ISO NA2] International Standards Organization, *Information Processing Systems - Data Communications - Addendum to the Network Service Definition Covering Network Layer Addressing*, ISO 8348/DAD2.

[ISO NLA] International Standards Organization, *Data Processing Systems - Data Communications - Internal Organization of the Network Layer*, ISO DP 8648.

[ISO NSD] International Standards Organization, *Information Processing Systems - Data Communications - Network Service Definition*, ISO 8348.

[ISO TPS] International Standards Organization, *Information Processing Systems - Data Communications - Transport Protocol Specification*, ISO 8073.

[ISO TSD] International Standards Organization, *Information Processing Systems - Data Communications - Transport Service Definition*, ISO 8072.

[ISO 84A] International Standards Organization, *A Formal Description Technique Based on an Extended State Transition Model*, Report ISO/TC97/SC16/WG1 NXXX, March 1984, Subgroup B on Formal Description Techniques (describes what is becoming known as the Estelle language).

[ISO 84B] International Standards Organization, *Definition of the Temporal Ordering Specification Language LOTOS*, Report ISO/TC97/SC16/WG1, May 1984.

[ISO 85] International Standards Organization, *Methodology for Protocol Testing*, Report ISO/TC97/SC16.

[JAFF 84] Jaffe,J.M., "Algorithms for Finding Paths with Multiple Constraints," *Networks*, Spring 1984.

[KAHN 78] Kahn, R.E., S. A. Gronemeyer, J. Burchfield, and R. C. Kunzelman, "Advances in Packet Radio Networks," *IEEE Proceedings*, November 1978.

[KEE 84] Intellicorp, *KEE - Knowledge Engineering Environment*, Palo Alto, CA.

[KOK 84] Kok, A.K, *The Decomposition of Real-Time-System Requirements into Process Models.*

[KOOM 85] Koomen, C.J., "Algebraic Specification and Verification of Communication Protocols," *Science of Computer Programming* 5 (1985): 1-36.

[KRAM] Kramer, B., "Stepwise Construction of Non-Sequential Software Systems Using a Net-Based Specification Language," *Advances in Petri Nets*, INCS, Springer Verlag.

[KUNG 83] Kung, R., and N. Shacham, *An Algorithm for the Shortest Path Under Multiple Constraints*, SRI, November 1983.

[LAUE 79] Lauer, G., *Stochastic Optimization for Discrete-Time Systems*, MIT, October 1979.

[LOGR 84] Logrippo, L., D. Simon, and H. Ural *Executable Description of the OSI Transport Service in Prolog*, Report TR-84-15, Computer Science Department, University of Ottawa, August 1984.

[LUCK 84] Luckham, D.C., F.W. Von Henke, "An Overview of ANNA, A Specification Language for Ada," *Conference on Ada Applications and Environments*, St. Paul, MN, October 1984: 116 -127.

[LUCK 85A] Luckham, D., and D. Helmhold, "TSL: Task Sequencing Language," in *Proceedings of the International Ada Conference - Ada in Use*, (Paris, May 1985), Cambridge University Press, 1985.

[LUCK 85B] Luckham, D., personal communication on a project to use TSL to specify a communications protocol, 1985.

[MACG 82] MacGregor, W., J. Westcott, and M. Beeler, "Multiple Control Stations in Packet Radio Networks," *IEEE MILCOM*, October 1982.

[MASC 80] *Royal Signals and Radar Establishment, Mascot Manual*, U.K., 1980.

[MCQU 80] Mcquillan, J.M., I. Richer, and E. C. Rosen, "The New Routing Algorithm for the ARPANET," *IEEE Transactions on Communications*, May 1980. See also the papers by McQuillan et al., and BBN reports referred to in this paper.

[MCRU 85] McCrum, Bill, personal communication with Bill McCrum of the Canadian Department of Communications on the Results of the May CCITT Meeting, 1985.

[MERL 79] Merlin, P, and A. Segall, *A Fail-Safe Distributed Routing Protocol*, September, 1979.

[MONK 84] Monk, Bob, personal communication from Bob Monk at Mitre Bedford on a proposed use of SADT as a front end for generating embedded system programs.

[NBS] NBS has developed a translator from protocol specifications to C.

[PARK 85] Park, Bert, personal communication with Bert Park of Telecom Australia.

[PDC 85] Protocol Development Corporation, *Estelle Development System: Functional Description*, Brookline, MA, 1985.

[PROB 85] Probert, Bob, personal communication with Bob Probert of Ottawa University, 1985.

[RESC 71] Rescher N., and A. Urquhart, "Temporal Logic," Springer-Verlag, New York, 1971.

[ROSS 85] "Douglas Ross Talks About Structured Analysis," Interview, in *Computer*, July 1985.

[RUDI] Rudin, FSM translator applied to SNA - secondhand verbal report.

[SCHW 80] Schwartz, M., and T. E. Stern, "Routing Techniques Used in Computer Communication Networks," *IEEE Transactions on Communications*, April 1980.

[SMITH 85] Smith C.U., G.A. Frank, J.L. Cuadrado, "An Architecture Design and Assessment System for Software/Hardware Code Design," *IEEE DAC* 1985.

[SOFT 83] *Proceedings of the Fifth International Conference on Software Engineering for Telecommunications Switching Systems*, Lund, Sweden, July 1983.

[TAJI 77] Tajibnapis, W.P., "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network," *Communications of the ACM*, July 1977.

[VHSIC 83] Westinghouse Defense and Electronics Center, Systems Development Division, *VHSIC System Programming Aides, Volume III: Directed Graph Methodology Reference Manual*, 9RA8738, Baltimore, June 1983.

[WEGN 84] Wegner, P., "Capital Intensive Software Technology," *IEEE Software*, 1, 3, July 1984.

[WEST 82] Westcott, J., and J. Jubin, "A Distributed Routing Design for a Broadcast Environment," *IEEE MILCOM*, October 1982.

[WOLP] Wolper, P.L., *Synthesis of Communicating Processes from Temporal Logic Specifications*, Department of Computer Science Report, Stanford University. Undated.

[ZAVE 84] Zave, P., "The Operational Versus the Conventional Approach to Software Development," *Communications of the ACM*, 27, 2, February 1984.

[ZAVE 85] Zave, P., "A Distributed Alternative to Finite-State-Machine Specifications," *ACM Transactions on Programming Languages & Systems*, 7, 1 (January 1985): 10-36.

2.0 COMMON ADA IMPLEMENTATION OF OSI AND DOD TRANSPORT AND INTERNET PROTOCOLS

2.1 Introduction

2.1.1 Purpose

The purpose of this section is to provide a technical definition of a project to analyze, design, implement and demonstrate a common Ada implementation of the OSI and DoD Transport and Internet protocols.

2.1.2 Terminology

In this document, TCP/IP refers to the combination of the DoD Transmission Control Protocol and Internet Protocol. TP-4/CLIP refers to the combination of the ISO Transport Protocol (class 4) and the ISO Connectionless Internet Protocol (known as CLIP) currently being defined as a subprotocol of the Network Layer. For brevity, TCP/IP and TP-4/CLIP are often referred to simply as "the DoD" and "the ISO" protocols, respectively.

2.1.3 Objectives

This project has several related general and specific objectives. The general objectives are as follows:

a. Provide technical input to the decision-making process regarding adoption of the ISO protocols by DoD.

b. Provide a tested, technical approach for making the transition to the use of the ISO protocols by DoD and to help reduce the risk and cost of such a move.

c. Contribute to the technical capability of DoD networks to interwork with ISO networks.

Given that a common design appears feasible, as indicated in Section 2.2 of this specification, the specific objectives are as follows :

a. Investigate design issues associated with the following requirements:

(1) A common Ada interface is required for implementations of the DoD and ISO protocols, which would enable higher level software to use either of the protocols in as flexible a manner as possible, thereby paving the way for a low-cost, low-risk transition to the use of the ISO protocols.

(2) A common Ada software organization is required for the DoD and ISO protocols, making full use of Ada's packaging capabilities. The extent to which this is practically possible will provide a measure of the confidence which can be placed in the commonality of function of the two protocols. Furthermore, it is expected that the Ada packages emerging from the satisfaction of this objective will be useful for the implementation of gateways between DoD and ISO networks. The investigation should contrast the common organization approach with that of providing only a common user interface to otherwise completely independent Ada implementations of the two protocols.

(3) It should be possible to use the packages in the common design in a transport level gateway between DoD and ISO networks.

b. Provide high level designs reflecting the results of the above analysis.

c. Provide implementations and demonstrations of the designs. The implementations are to be in Ada.

Although this project is intended to produce Ada software for both the DoD and ISO protocols, the production of software for the individual protocols is not to be regarded as a primary objective, in isolation from the other objectives.

2.1.4 Scope

Considering all of its phases, the entire project is potentially very broad in scope, including design, implementation, test and demonstration of Ada implementations of several protocols, following a path which is almost certainly different from that taken by current implementors of DoD protocols in Ada. However, the scope can be reduced by eliminating phases to obtain less complete but still useful results.

Operating system issues are outside the scope of this project. Required is an embedded system design approach, which makes full use of Ada tasking and assumes no operating system constraints. This restriction is imposed for several reasons:

a. It will help to ensure that a general design approach is taken, without concern for the peculiarities of particular operating systems.

b. It constrains the issues addressed by this project to Ada-specific issues.

c. It is completely appropriate in itself for applications in which the communications software forms part of an embedded system, such as a gateway. This restriction is not intended to exclude the possibility of future integration of the resulting software into particular operating system environments.

Outside the scope of this project is the development of the lower level communication software required to perform actual communication tests over real networks. It is assumed that existing software can be used. Where such software is to be found or how it is to be organized is not defined in this document. However, wherever found or however organized, it should not dictate the design of the higher level software. Rather the use of the lower level software should be made possible by an interface which conforms to the design paradigms developed during this project for the higher level software. There is no requirement that the existing lower level communication software be in Ada.

Also outside the scope of this project is the alignment of the work of this project with other work which may be going on to implement the DoD protocols in Ada. This project is expected to take an independent path, dictated by the requirement for maximum commonality of organization between the DoD and ISO implementations.

2.1.5 Outline of this Document

Section 2.2 gives an overview of the networking models for the DoD and ISO protocols and then considers how the differences between them impact software organization.

Section 2.3 describes the general approach to be taken to design, implementation and test.

2.2 Background

2.2.1 Introduction

A recent National Research Council study [NRC 85] drew the following conclusions about the DoD's TCP and the ISO's TP-4:

a. They are functionally equivalent.

b. They provide essentially similar services.

c. Neither is technically superior from any point of view, including security.

d. New applications can be programmed with similar levels of effort to use either one.

e. TP-4 will meet military requirements.

Advantages of using TP-4 include compatibility with NATO, accessibility to commercial software and services, interoperability with a wider community of users and systems, and lower life cycle costs.

The study noted that networking plans underway in DoD now (of which WIS is a major component) imply a very high cost for moving to TP-4 later, because by the late 1980's, DoD networks would be virtually all TCP-based.

The committee recognized that a change from TCP/IP is viewed with "some trepidation" in DoD, but nevertheless recommended that it take place at some time.

This section first reviews the networking models for the DoD and ISO protocols and then considers how the differences between them impact software organization.

2.2.2 Networking Models for DoD and ISO Protocols

2.2.2.1 Introduction

This section describes the underlying models which form the basis for the DoD and ISO Transport and Internetwork protocols. These models are presented in the form of block diagrams, and identify interacting entities and the nature of the interactions.

The purpose is to arrive at a unified conceptual framework for a common implementation of the protocols and to consider the constraints, if any, that are imposed by this common framework.

2.2.2.2 The Internet Model

2.2.2.2.1 The DoD IP Model

The term "catenet" is commonly used to refer to the collection of packet networks interconnected by means of the Internet Protocol.

The catenet is assumed to have the following characteristics:

a. While various technologies may be used in the individual networks, end-to-end information exchange is achieved using internetwork datagrams. The capability to transfer datagrams must therefore be built into or on top of each local network.

b. Each network supports the transfer of datagrams containing no less than 1000 bits of user data; this does not restrict the type of network that exists in the catenet; virtual circuit and circuit switching networks are also permitted in addition to pure datagram networks if they permit datagrams to be carried and if the switching time is sufficiently fast.

c. Datagrams may be lost, duplicated, or delivered out of sequence.

d. Networks are interconnected via catenet gateways. A gateway is logically viewed as consisting of two "gate-way halves." Each half-gateway has two interfaces, one to a local network, and the other to another gateway half. Gateways which are visible to the catenet model have the characteristic that they can interpret the address fields of internet datagrams so as to route them to other gateways or to destinations on the directly attached network.

e. Local gateways are associated with each host; they perform reassembly of fragmented internet datagrams, encapsulation of internet diagrams in local network packets, and routing of datagrams from the host to internet gateways.

f. No explicit network hierarchy exists; every network is known to all catenet gateways and each catenet gateway knows how to route internet datagrams so they will eventually reach a gateway connected to the destination network. This assumption leads to a flat internet address space (see the discussion of addressing in Section 2.2.3.4).

Figure 1 illustrates the basic DoD IP Model.

2.2.2.2.2 ISO Network Layer Model

The networking model developed by ISO [ISO NLA] is somewhat more general than the DoD one. It supports both connection-oriented and connectionless data transfer whereby connectionless corresponds to datagram operation. The model applies both to networking within a single subnetwork and to networking across many different subnetworks. Only the subnetwork interconnection aspects of the model are considered here.

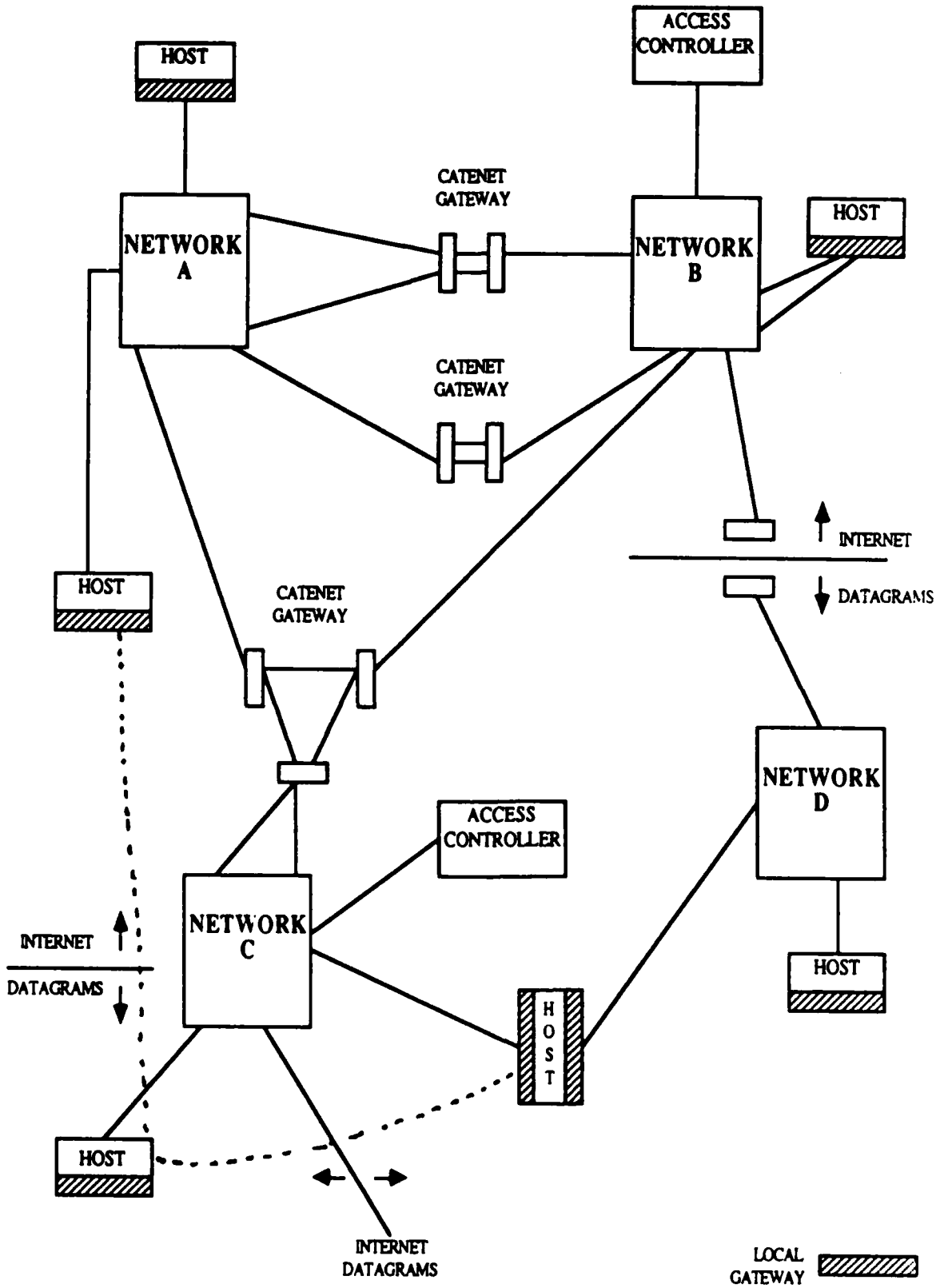


Figure 1. Basic INTERNET Model

Four different approaches to the interconnection of different subnetworks are considered:

a. Interconnection of OSI Subnetworks: In this case, each subnetwork is an OSI Subnetwork, i.e., it fully supports the OSI Network Service, as defined in ISO Draft International Standard 8348 [ISO NSD] and its first Addendum [ISO NA1]. As a result, only a single Network Layer protocol is used within each subnetwork, and interworking units attached to adjacent networks perform the necessary relaying and routing functions to support the OSI Network Service end-to-end.

b. Hop-by-Hop Harmonization: This approach involves a network environment which includes at least one subnetwork which does not conform to the OSI Network Service. In this case, each such subnetwork is made to conform by adding one or more convergence protocols which mask or enhance the subnetwork services as necessary. When all subnetworks have been made to conform to the OSI Network Service, then they are interconnected via interworking units which perform routing and relaying functions as above.

c. Internetwork Protocol Approach: In this case, the OSI Network Service is provided by defining a protocol which operates across the set of interconnected subnetworks. This protocol is subnetwork independent and is called a "subnetwork independent convergence protocol (SNICP)". ISO has defined such a protocol [ISO CLIP]; it is referred to in this report as the ISO Connectionless Internet Protocl (CLIP). For it to operate over a variety of subnetworks, it must rely on a defined set of supporting capabilities. These capabilities may or may not be provided by a particular subnetwork's internal protocol. In that case, another protocol is required to make the subnetwork conform to the requirements of the SNCIP. This additional protocol is called the "subnetwork dependent convergence protocol (SNDCP)".

d. Combination of Approaches: In this case, some combination of the first three approaches is used to achieve the internetwork service. If an internetwork protocol is used in this environment, it does not act over all networks, but acts as a "multi-hop" convergence protocol.

Figure 2 provides a view of the organization of the Network Layer within an interworking unit connected to two subnetworks. This figure shows all possible roles that may be played, but depending on circumstance, either or both of the convergence protocols may be omitted.

Figure 3 shows the interconnection of two connectionless subnetworks using a connectionless internet protocol. In this diagram, each end system implements all of the three Network Layer protocols, i.e., the Subnetwork Access Protocol (SNAP) to gain access to the local subnetwork, the SNDCP to enhance the Subnetwork Protocol to suit the needs of the Internet Protocol, and the SNICP which is end-to-end in nature. (NOTE: While the SNICP is identical in both networks, the SNDCP and SNAP may be different.)

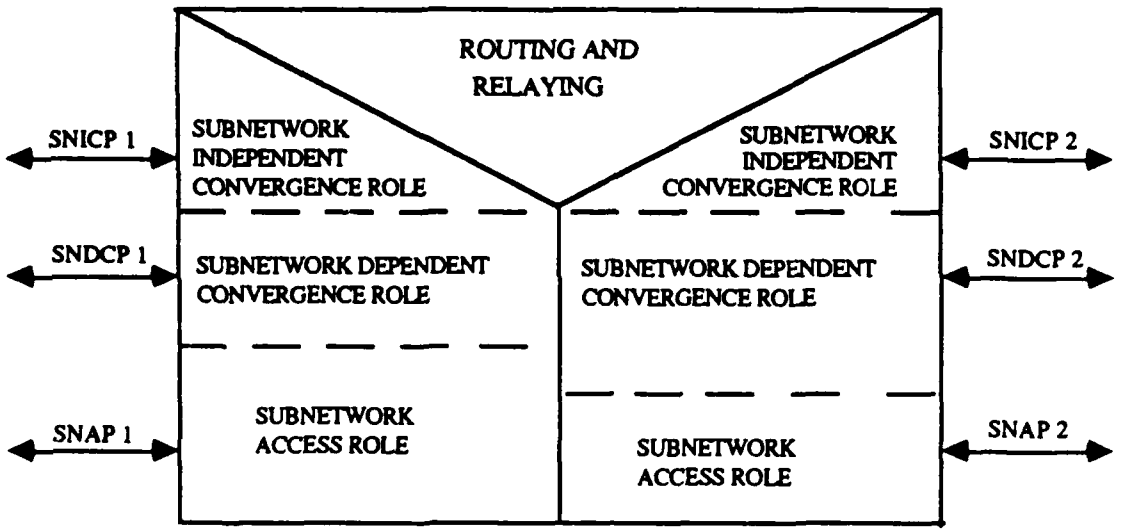


Figure 2. Organization of a Network Layer Interworking Unit

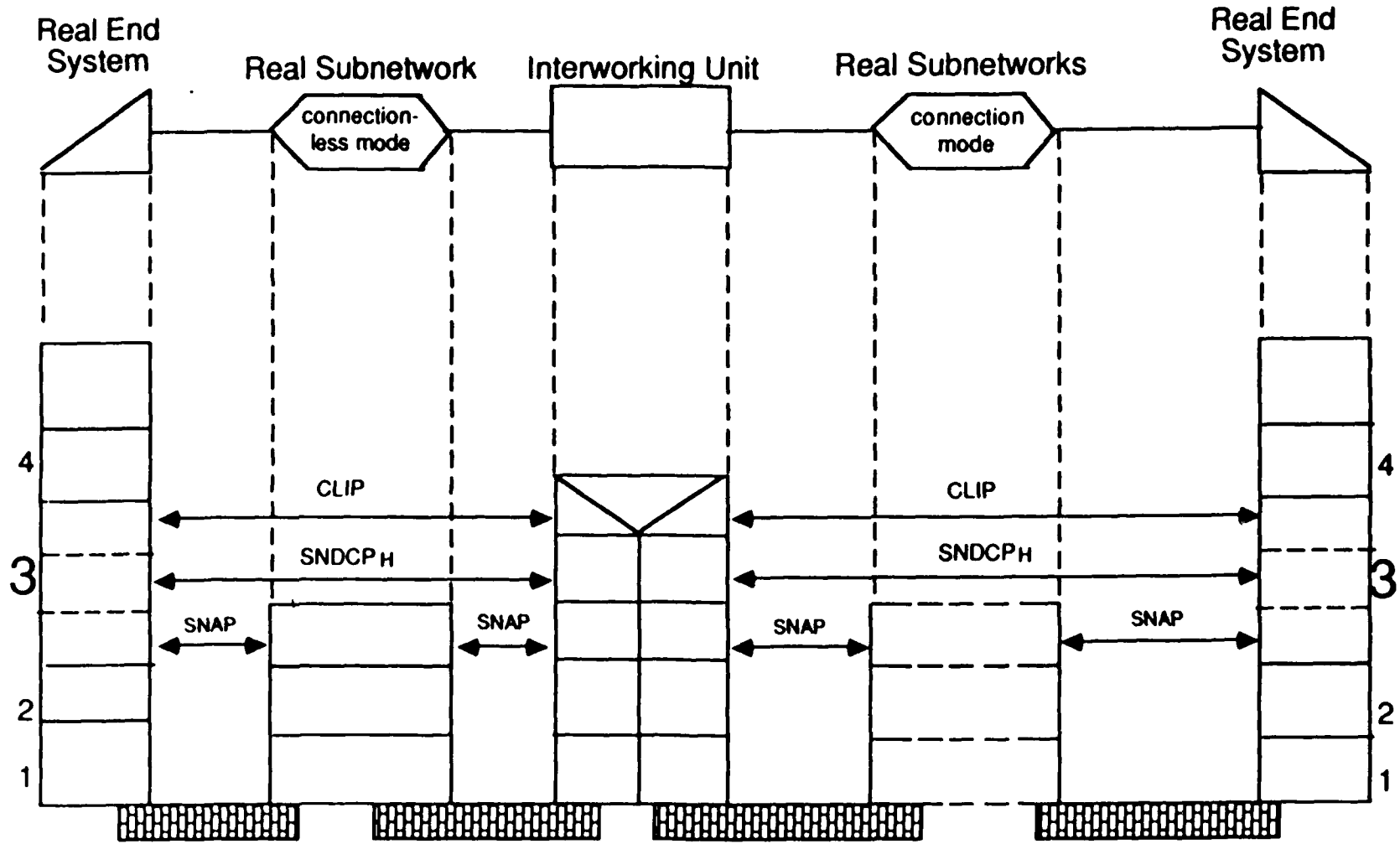


Figure 3. Interconnection of Connectionless Subnetworks via a Connectionless Internet Protocol

2.2.2.3 Relationships

The DoD model of internetworking is accommodated within the ISO view of the Network Layer and hence, there is no conceptual conflict between the two. When one equates the IP protocol to the ISO SNICP and gateways to interworking units, one arrives at the same view of interworking as that portrayed in Figure 3. Although there is no explicit mention in the DoD literature of a SNDCP, the potential need for such a protocol is implied in the statement that any local network may be used with IP as long as it provides a datagram transfer service. This requirement may in some cases call for the addition of an additional protocol to handle datagram interchange.

To further verify consistency of the two approaches, it is necessary to consider the nature of the services provided by the DoD and ISO Network Services. As already noted, the DoD Network Service is a simple datagram one, with no provision for reliable end-to-end transfer of datagrams. Similarly, the ISO Connectionless Service [ISO NA1] provides a single primitive group for data flow, namely UNITDATA. As with the IP datagram, there is no guarantee that data supplied to the Network Service will not be lost, duplicated or corrupted by the Network Service Provider. (NOTE: Other primitive groups are defined for conveying service characteristics, but these are related to layer management.)

We may conclude then that there is a unified framework for a common implementation of the DoD IP and the ISO Internetwork Protocol. This framework is constrained to a connectionless Network Service with the use of an internetwork protocol to permit interworking among diverse networks.

2.2.2.3.1 The DoD Transport Model

The internetwork environment in which the DoD TCP protocol is to be used consists of hosts connected to networks which are in turn interconnected via gateways. The active agents that produce and consume messages are processes. In order to perform their tasks, processes may need to communicate with processes on other hosts. The primary purpose of the TCP is to provide reliable, securable logical connection service between pairs of processes in the face of an unreliable supporting datagram transfer service.

As the TCP operates between processes located in hosts, the protocol is end-to-end in nature. Figure 4 illustrates the DoD transport model.

2.2.2.3.2 The ISO Transport Model

The ISO Transport Service [ISO TSD] provides transparent connection-oriented transfer of data between Transport Service Users. It relieves these users from any concern about the detailed way in which supporting communications media are utilized to achieve this transfer.

Five classes of Transport protocol [ISO TPS] are defined to cater for different network characteristics and user requirements. Of these five classes, Class 4 is of primary interest, as it is intended for use over unreliable networks, and can be used with either a connection-oriented or connectionless network service. Among other facilities, this protocol class provides mechanisms for the detection and recovery from lost, duplicated, out-of-sequence or damaged protocol data units.

Figure 5 illustrates the environment in which the Transport Class 4 can operate.

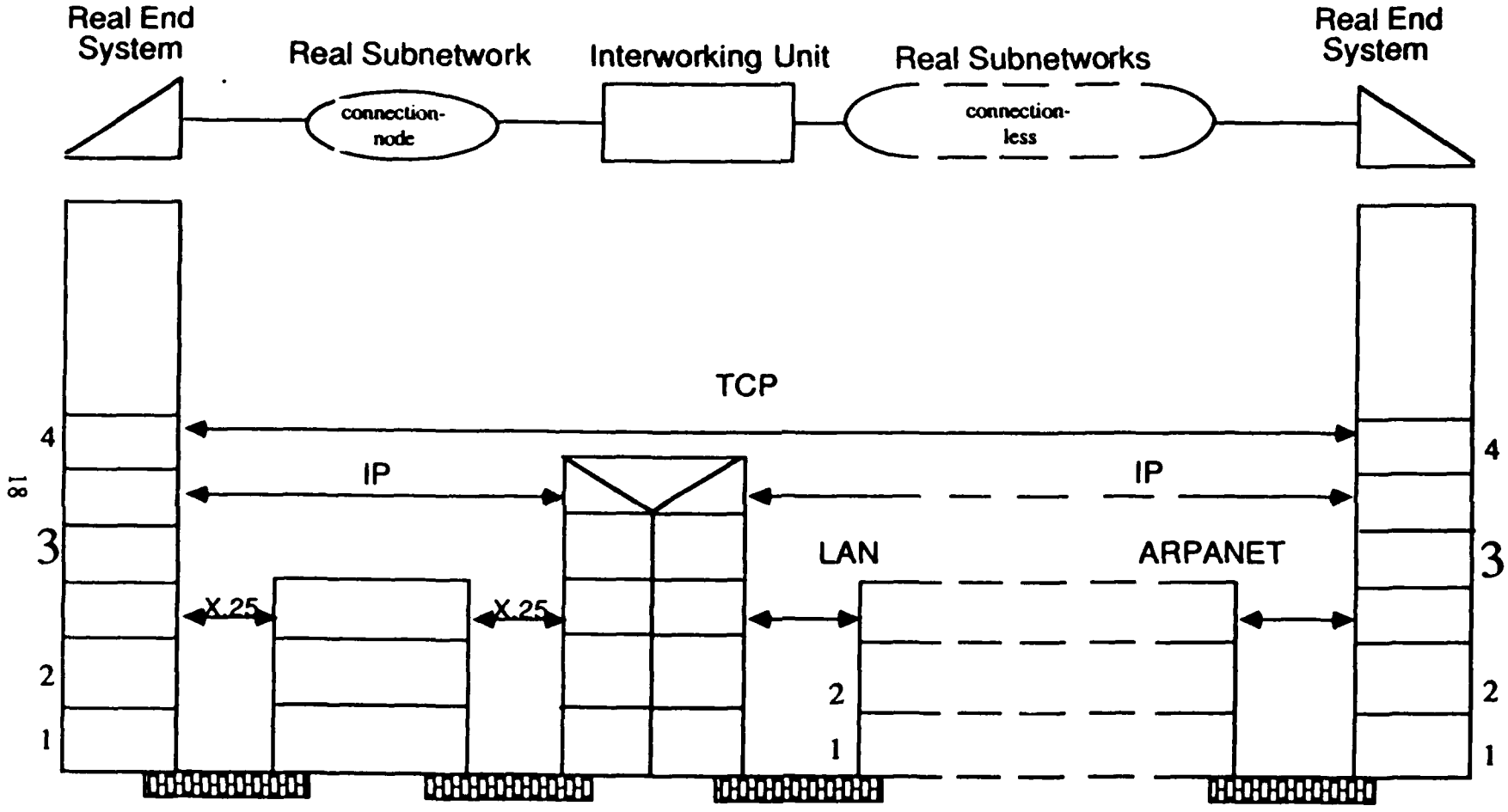


Figure 4. DoD Transport Model

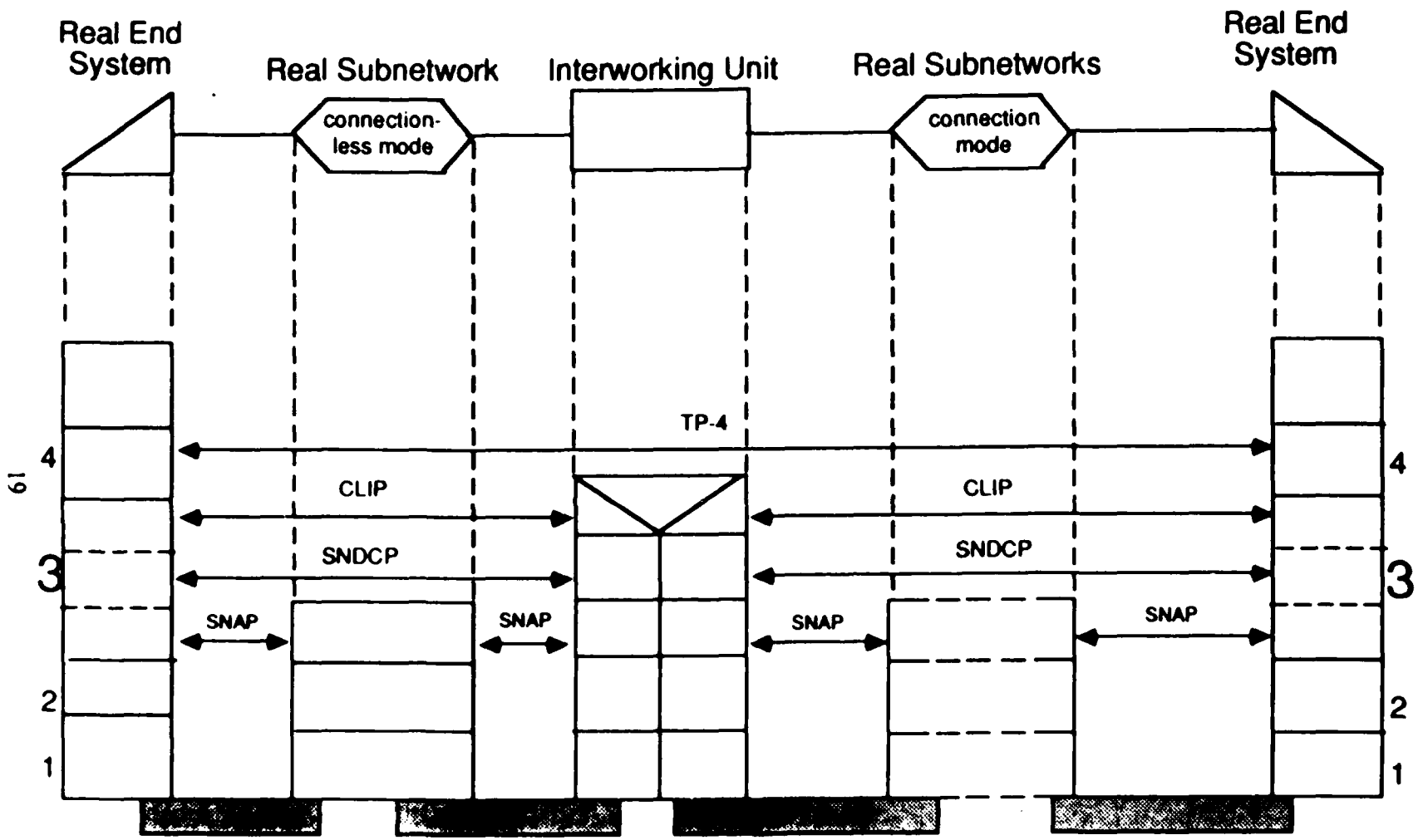


Figure 5. Environment of ISO TP-4, Showing its Use With Connection-Oriented and Connectionless Networks

2.2.2.3.3 Relationships

As indicated by Figures 4 and 5, the environments of the DoD and ISO Transport services overlap. The ISO Transport Service is more general in that it provides for five different protocols and can operate in both connection-oriented and connectionless modes.

We may conclude then that a unified architectural framework exists for a common implementation of the DoD and ISO Transport Protocols. This framework is constrained to reliable, connection- oriented transfer of user data over a connectionless network service.

2.2.3 Impact of Service and Protocol Differences on Software Organization

The previous section has demonstrated that a framework exists for the development of common implementations of the TCP/TP-4 and IP/CLIP protocols. However, there are many issues that will affect the degree to which commonality of implementation is possible. Two of the most important issues relating to a common implementation is the possibility of a common upper layer interface and the possibility of a common internal organization of the protocol software within a layer or sublayer. These issues are related to the similarities and differences in the services and protocols defined by DoD and ISO. This section identifies some of these similarities and differences, based on analyses by [NRC 85] and [BOCH 84], and where appropriate considers the potential impact on a software implementation. A designer may wish to consider other implications.

2.2.3.1 Transport Services

The ISO Transport Service Definition [ISO TSD] provides an abstract definition of the services available to the Transport Service User. On the other hand, the TCP specification [DoD TCP] provides a more concrete description of the interface between TCP and the user, in the form of procedure calls. The following discussion will be in terms of service features, and will identify differences in the user's view of the Transport service for TCP and TP-4.

2.2.3.1.1 Call Collision

When an outgoing call collides with an incoming call, with TCP, only one connection results, while with TP-4, two connections may result. This difference should have a minor impact on software organization. Most distributed applications involve a master-slave relationship, such that one particular site would typically initiate connections while another would typically wait for incoming calls.

2.2.3.1.2 Multiple Connections Per Service Access Point Pair

TP-4 permits many simultaneous connections for a given pair of Transport addresses; each connection is distinguished via connection endpoint identifiers. TCP identifies a connection uniquely on the basis of a pair of sockets corresponding to the communicating partners. Hence, two connections cannot be established simultaneously between two sockets. The problem of creating two TCP connections between the same pair of endpoints can be resolved by having the initiating TCP entity assign new TCP ports (and hence new sockets) as needed.

2.2.3.1.3 Addressing

TCP uses a fixed address length while ISO is adopting a hierarchical, variable length addressing scheme for its protocols, with the Transport address consisting of the Network address plus an optional suffix. The implication of this for software is the need for a flexible parameter passing mechanism which supports variable length addresses.

2.2.3.1.4 User Data During Connection Establishment

This is permitted by TP-4 but not by TCP. This issue has little impact on software organization, but would affect user processes that would like to user both TCP or TP-4 in a consistent manner.

2.2.3.1.5 Quality of Service Selection

TCP allows the user to specify the following quality of service parameters: precedence, security/compartment and data delivery timeout interval. TP-4 provides for the following: connection establishment delay, throughput, transit delay, residual error rate, connection release delay, protection, and priority. It also provides for monitoring and reporting other quality of service parameters, such as probability of connection establishment failure, probability of connection release failure and resilience. This issue has little impact on software organization, apart from a requirement to support all possible parameters.

2.2.3.1.6 TSDU Delimitation

TCP provides octet stream data transmission with a "push" feature to indicate that all outstanding data is to be transmitted at once. No particular semantic is associated with the "push", although it is preserved on an end-to-end basis. TP-4 has the concept of a TSDU, a unit of user data that is to be preserved on an end-to-end basis. The setting of the end-of-TSDU flag by the sending user has no explicit protocol implications, although it seems to imply transmission of all preceding data. Thus, although the mechanisms are slightly different, the same interpretation can be associated with both; in which case, there is no software organization issue.

2.2.3.1.7 Out-of-Band Signals

From a service perspective, the expedited data option of TP-4 and the "urgent" flag of TCP are equivalent provided the user is not interested in preserving the relationship between the urgent data and the normal data flow. TCP preserves that relationship while TP-4 does not.

2.2.3.1.8 Orderly Release

TCP supports both orderly and abrupt connection release while TP-4 supports only abrupt release, although a similar service is available from the Session layer. Bochmann [BOCH 84] suggests that this discrepancy can perhaps be resolved by considering the TCP protocol as providing Transport and some Session functionality, with the orderly release service being considered as a Session service. In this way, consistency of architecture can be maintained.

2.2.3.2 Transport Protocol

2.2.3.2.1 Error Detection

Both TCP and TP-4 provide protection against damage to protocol data units via a checksum algorithm. The algorithm is different in each case, but the level of protection appears to be equivalent. This difference should have little impact on software organization.

2.2.3.2.2 Flow Control

TCP provides flow control via a credit in terms of octets and uses variable length data segments, while TP-4 handles credit in terms of fixed-length protocol data units (the size of which is negotiated during connection establishment). These approaches suggest different buffer management approaches; this issue will require careful study if a common implementation is to be achieved.

2.2.3.2.3 PDU Syntax

The syntaxes of the TCP and TP-4 protocol data units (PDU) are different, but this should not have a large impact on software organization.

2.2.3.3 Common Transport Service

Evaluation of the functionality of TCP and TP-4 [BOCH 84] indicates that there is sufficient commonality in the connection establishment, normal data transfer, urgent data transfer, and abrupt connection termination services as a basis for a common implementation of these services with a shared user interface. Other services, such as orderly connection release, may have to be considered as outside a common implementation.

2.2.3.4 Network Services

The DoD IP and the ISO CLIP are much more similar to one another than the transport protocols. This results from the fact that the ISO IP was derived from the DoD IP. There are nevertheless some service differences, although none are very significant.

Of the service primitives defined in the ISO Connectionless Network Service Definition [ISO NA1], namely UNITDATA request and indication, FACILITY request and indication and REPORT indication, only the UNITDATA and REPORT primitives are supported by the ISO CLIP. The latter primitive serves to report network errors.

The DoD IP defines only the equivalent of the UNITDATA primitives, but a companion protocol, the Internet Control Message Protocol (ICMP) is defined to handle error and management facilities. This protocol is outside the scope of this project.

Apart from error reporting, the other major significant difference between the DoD and ISO services is addressing. The DoD IP has fixed length, 32-bit source and destination addresses (identifying network and host) plus an 8-bit "protocol number" filed to identify the higher-level protocol for which the IP data is intended. The ISO addressing scheme, as defined in [ISO NA2] is based on the concept of hierarchical addressing domains. This concept divides the global addressing space into a set of distinct domains, each of which may be subdivided into subdomains, and so on. Each subdomain may have its own addressing scheme. A hierarchical network address is structured so that, at any level of the

hierarchy, an initial part of the address unambiguously identifies a subdomain, and the rest is allocated by the management of the subdomain to unambiguously identify either a lower level subdomain or a network service access point within the subdomain.

The effect of this addressing scheme is that network addresses are variable length. The generality of this scheme permits the inclusion of the DoD IP addressing scheme as a valid subdomain addressing scheme, although the DoD scheme would require extension to incorporate an Initial Domain Part (IDP) if it is to be used with the ISO CLIP.

The "protocol number" field of the DoD IP has no direct counterpart in the ISO addressing scheme.

2.2.3.5 Network Protocol

Functionally, the DoD IP and the ISO CLIP are very similar, with the major protocol difference being different Protocol Data Units (PDU) formats. The ISO CLIP PDU format, while based on the DoD IP PDU format, has different field naming, ordering and coding. From a software organization viewpoint, there is no significant difference between the two protocols.

2.3 Approach

This section defines in general terms the approach to be taken to the design, implementation and test activities.

2.3.1 Candidate Designs

The analysis should take account of all points raised in Sections 2.1 and 2.2 to arrive at a small set of candidate alternative software designs for both stand alone systems and gateways.

2.3.2 Software Organization

This project aims at design issues which are Ada-specific and operating system (OS) independent. By the term "operating system" we mean a traditional, general purpose operating system which requires an organizational difference between "application" software above the OS interface and "system" software below it. The design approach is not to be constrained by issues associated with integrating the transport and internet software into such an operating system environment. For example, an approach where the transport and internet software is designed as system programs to fit into the "driver" framework of such an operating system is inappropriate. Also inappropriate is an approach where the transport and internet software is designed as application programs of such an operating system, tailored to the limitations of its application program interface.

The requirement is to devise a set of Ada packages and an appropriate control framework, including appropriate use of Ada tasking, such that a DoD or ISO version of the software can be obtained simply by "plugging in" the DoD or ISO version of each package into the framework using "with" before compilation. There is no requirement to have a combined implementation with dynamic selection of DoD or ISO services at run-time. The intent is that this configurability extend to the common interface presented to the higher layers.

In order to accommodate the stream-oriented nature of existing higher level DoD software which now uses TCP, a stream-oriented interface to the TP-4 version of the common design is desirable. To accommodate the block-oriented nature of ISO higher level protocols (e.g., session), a block-oriented interface to the TP-4 version of the common design is desirable. Therefore it will be necessary to provide the choice of configuring the software with a stream oriented or a block-oriented interface. The issue of whether or not to provide a block-oriented interface to TCP in the implementation is left as an open question to be resolved. The possibility of doing so was raised in the NRC report and is implicit in the design approach described here.

The commonality of the design for the DoD and ISO protocols is to extend down to as fine a level of granularity as is logically possible. Only below this level of granularity will the organization of bodies differ. A single package with everything organized differently in the body will not satisfy the requirement.

This document does not give a design structure. It is left to the implementor to devise a good structure, following good design principles, such as those enunciated in [BUHR 85].

2.3.3 Implementation and Test

The implementor will be responsible for defining an appropriate implementation and test environment, subject to the other requirements and constraints given in this specification. The requirement is for a system containing the appropriate Ada transport and internet programs, together with any lower level software (not necessarily in Ada) required to support network access.

The Ada compiler and run-time system must be selected with particular attention paid to run-time efficiency of tasking.

Although the design approach has been characterized earlier as an "embedded system approach", this does not necessarily require that the test environment be a stand-alone, special purpose, embedded system, although this is one possible alternative.

2.3.4 Demonstration

The demonstration has three purposes:

a. Provide visible evidence of conformance of the protocol implementations with the corresponding specifications.

b. Show interoperability, i.e., ability to communicate with other implementations of the protocols.

c. Show interworking between a DoD network and an ISO network via a TCP/TP4 gateway.

d. Provide performance metrics (throughput and response time).

2.3.4.1 Conformance Tests

Evidence of conformance to the protocol specifications is provided by exercising the implementations with established (where possible) test suites. This will require demonstrating communication between systems implemented by the designer using the test suites.

In the case of TP-4, the existing NBS test suite may be used. In the case of ISO CLIP, the NBS test suite currently under development for AUTOFACT 85 may be used. In the case of TCP/IP, the implementor should use whatever DoD test suites are available by the time of the testing, and to develop its own otherwise, based on the conformance testing methodology described in [ISO TEST].

These conformance demonstrations will consist of a physical demonstration of basic interconnection tests using a test configuration appropriate to the test suite in use (e.g., the distributed test configuration used with the NBS ISO IP test suite), and of written evidence of successful completion of all other tests.

It will be necessary to demonstrate conformance of each protocol individually and also to demonstrate conformance of TCP operating in conjunction with DoD IP and TP-4 operating in conjunction with ISO CLIP.

2.3.4.2 Interoperability Tests

This aspect of the demonstration will verify that the four protocol implementations can operate correctly with existing implementations performed by other agencies.

At least two other different implementations should be included in these tests, one acting as gateway or interworking unit, and the other acting as a host or end system on a different subnetwork. The choice of test "partners" and subnetworks for these tests will be at the implementor's discretion.

Wherever possible, the "foreign" implementations should include both the transport and internet protocol pairs.

2.3.4.3 Gateway Demonstration

The implementor should demonstrate a gateway implementation, using the packages already developed to satisfy the other requirements of this document. The gateway will interconnect a DoD network with an ISO one, allowing the common subset of services identified in Section 2.3.3 to operate on an end-to-end basis. This use of existing packages to implement this gateway implies that the gateway operates at the service interface level as discussed in [BOCH 84].

2.3.4.4 Performance Demonstration

The implementor should measure the performance of each implementation and compare with metrics available for other implementations. The specific performance parameters to be measured are throughput and response time. Other parameters may be included at the implementor's discretion.

3.0 TOWARDS AUTOMATIC GENERATION OF ADA PROTOCOL SOFTWARE FOR WIS

3.1 Introduction

3.1.1 Purpose

The general problem addressed by this section is the automatic generation of Ada protocol software for WIS from protocol specifications. This document describes two projects which are intended to establish a technical baseline for proceeding later to a prototype system for this purpose. The two projects, which may be pursued in parallel, involve gaining experience generating Ada code fragments from protocol specifications and comparing alternative approaches to solving the combined environment problem. The prototype system itself is outside the scope of these projects; it will be the subject of a future report to be prepared by the design authority once the technical baseline has been established.

The issues to be addressed by the projects defined in this section arise from the fact that protocol specifications are not by nature sufficient to define complete software implementations. This is not a shortcoming but rather a deliberate property, needed so specifications can be prepared and analyzed without concern for implementation details. Consequently, code generated automatically from protocol specifications without further software design information is inherently fragmentary. The fragments are neither compilable nor executable as complete programs. To solve the general problem, not only are protocol specifications needed, but also program design specifications. Information required in the program design specifications includes such items as the desired module and control structure of the program (e.g., in terms of Ada packages, procedures and tasks) and its data structure (e.g., in terms of abstract data types encapsulated in Ada packages).

This need to combine protocol specifications, software design specifications, and the Ada language raises issues which have been incompletely resolved in the literature. The projects described here are intended to resolve these issues and to establish a baseline for later prototyping work.

Section 3.2 provides background on the major issues and the current work in the areas relevant to this project.

Section 3.3 provides a preliminary set of requirements for the ultimate combined environment and describes the approach to be taken to achieve the short term objectives in relation to this set of requirements.

3.1.2 Objectives

3.1.2.1 Long Term Objectives

The long term objective is the development by the 1990's of a combined protocol software design environment capable of taking protocol specifications and program design specifications as input and producing complete, compilable, executable Ada programs for WIS protocols as output.

For brevity, this objective is sometimes indicated in this document using the phrases "combined environment" and "solving the combined environment problem".

Requirements defined in Section 3.3 for the combined environment provide guidelines for the approach to be taken in the baselining projects.

3.1.2.2 Short Term Objectives

The following two short term objectives provide the basis for a two-pronged attack on establishing a baseline for future prototyping work.

a. First short term objective: Ada-specific experience with code fragment generation

   The first short term objective focuses on Ada and protocols, without tackling the combined environment problem. It is to gain insight into the problems involved in generating complete, compact, efficient Ada code fragments from protocol specifications. The intent is not to break new ground from a protocol or an environment viewpoint, but rather to gain insight into Ada-specific problems by extending work already done in the protocols community to include Ada.

   While the objective does not focus on the combined environment problem, there are lessons to be learned in pursuing this objective which affect the solution of that problem. Therefore, a secondary objective is to extract these lessons, by analyzing how the generated code fragments could be integrated manually into complete programs.

   The emphasis is on gaining insight by experiment into Ada-specific problems and issues rather than on producing reusable code generator software.

b. Second short term objective: comparison of selected candidate approaches to solving the combined environment problem

   The second short term objective is to gain insight into the strengths and weaknesses of different approaches to solving the combined environment problem to meet the requirements identified in Section 3.3. Several selected approaches which must be investigated are identified later. However, these are only to provide a starting point and are not intended to prevent the designer from proposing other approaches for investigation. The aim is to provide a clear set of choices to the design authority for approaches to the prototyping work, together with all the information necessary to evaluate the choices and to specify a prototyping project.

3.1.3 General Approach to Pursuit of the Objectives

The general approach required to the pursuit of the short term objectives is outlined here. More details are provided in Section 3.3.

The two short term objectives are to be pursued in parallel. Benefits of doing so are early insight and experience on a broad front. This should result in the fastest progress toward achieving the long term objectives.

The first short term objective, Ada-specific experience with code fragment generation, is to be pursued by repeating the intent of selected experiments in automatic code fragment generation performed in the protocols community for other target languages than Ada, taking due account of both similarities and differences between Ada and the other

languages. The intent is not to repeat the experiments precisely, because in many cases this would be uninteresting. For example, producing Pascal or C code in Ada syntax for experiments where the original target language was Pascal or C would not give much insight. However, experiments in generating CHILL code may give direct insight into what kind of Ada code to generate, because of similarities between Ada and CHILL. The emphasis is on learning how to generate the best possible Ada target code, taking advantage of unique features of Ada, such as packaging, tasking, generics and exceptions. Criteria of goodness for the target code are included in the requirements defined for the combined environment in Section 3.3.

The second short term objective, comparison of selected candidate approaches to solving the combined environment problem, is to be pursued as a paper study, taking account of the literature and of direct contact as necessary with selected researchers and laboratories to get the current information. Included in the scope of this work is:

a. Preparation of a comprehensive annotated bibliography of relevant work

b. Detailed definition of alternative approaches

c. Detailed definition of combined environment requirements

d. Analysis of the strengths and weaknesses of the various approaches relative to the requirements

Sections 3.2 and 3.3 of this document provide a starting point for this work. This work should in total present a clear set of choices to the design authority for approaches to the prototyping work together with all the information necessary to evaluate the choices and to specify a prototyping project.

3.1.4 Scope

The baselining projects are aimed at addressing Ada-specific and protocol-specific issues. Operating system issues and Ada run-time system issues are outside of their scope. An approach which is independent of these issues is required. The designer should assume protocol software can use all of the features of Ada to form any desired program organization. For example, there is no need to be concerned about forcing protocol software into arbitrary device driver formats for existing operating systems, which in some cases would preclude the use of many Ada features, particularly tasking. This is compatible with the WIS local area network (LAN) operating system (OS), because it is assumed that:

a. WIS OS will allow full Ada above it.

b. All the protocol software considered for automatic generation will be above WIS OS.

The WIS OS should be viewed as a run-time system for the execution of Ada programs in a distributed fashion on a LAN.

Also outside the scope of the baselining projects are methods for analyzing and testing protocol specifications themselves. The concern here is only the successful conversion of protocol specifications into programs. Current protocols research should produce solutions to the problem of testing the protocol specifications themselves.

Outside the scope of the baselining projects is the detailed specification of a combined environment to be prototyped. The concern here is to provide enough information for selection of a prototyping approach and specification of a prototyping project, without requiring further research.

3.2 Background

3.2.1 Introduction

This section summarizes the relevant state of the art and identifies the key issues in the combination of protocol specifications, software design specifications and the Ada language.

There are two major streams of world-wide research activity which affect this project:

a. Formal techniques for protocols: This activity includes formal specifications for communication protocols and their translation into implementations.

b. Software design environments: This activity includes operational software design [ZAVE 84] and rapid prototyping. Because protocol implementations need to deal with real-time events and concurrency, they are viewed in this document as falling within the class of embedded systems. Accordingly, the most relevant work for this project is on software design automation for event-driven, real-time, embedded systems.

Techniques from both of these areas will be needed to solve the general problem.

These areas, especially when considered in combination, are going through a rather explosive and chaotic period of development at present. There are many competing techniques and approaches being proposed and tried to solve aspects of the overall problem and very few which even attempt to tackle the overall problem head on. It is too early to choose a single "best" way of proceeding to achieve the long term objective. However, what can be done now is to analyze the existing work and extract the major thrusts which look most promising. This section makes a start in this direction which the baselining projects are intended to complete.

The mandated use of Ada for this project provides an interesting overlay on all of this activity. While Ada is a natural language for implementing protocols [BUHR 84], there is not universal agreement about its appropriateness for implementing environments to automate the software life cycle. [WEGN 84] Furthermore there is little evidence of experimental work in this area which uses Ada as the implementation language. [FREN 85] According to its detractors, Ada is too rigid and too rooted in old software technology for this purpose. Experience has shown that languages like Prolog are more suitable for implementing experimental environments because of the speed with which ideas can be prototyped. [BUHR 85B] The baselining work is expected to shed more light on this issue.

3.2.2 Terminology

From a communications perspective, the term protocol refers to the interchange of messages flowing between independent computer systems (the nodes of the computer network), usually over serial communication paths spanning two or more computers. Such communication protocols are often organized hierarchically into protocol suites; messages of higher level protocols may be enveloped in those of lower level ones. Furthermore, they often operate in temporal phases during which different rules apply. From this perspective, a protocol specification is the set of rules governing the formation and sequencing of the inter-node messages, taking account of both levels and phases.

Because the term protocol is used in two different senses in this document, definition of terms is required. The distinction between "peer" and "local" protocols must be made. Local is not used here in the sense of local area network, but in a different sense which is explained later. Usually it is only the peer protocols that are standardized; the local protocols are largely left to the implementors.

Inter-node protocols maybe referred to as "peer" because they are between entities at the same logical level. When communication specialists refer to protocol specifications, they are usually referring to the peer protocols. Protocol standards such as those of the International Standards Organization (ISO) and Consultative Committee for International Telephone & Telegraph (CCITT) are concerned only with standardizing the peer protocols. Although such standards are increasingly being phrased in terms of so-called reference models which seem to refer to the internal organization of the nodes implementing the protocols, in fact there is no standardization implication for the internal organization. ISO terminology includes suggestive terms such as protocol entities, service primitives, modules, channels and so on. The reference model terminology is used only with the intent of enhancing clarity of exposition.

The nodes of the computer network contain "local" protocols governing the interactions among the software modules which implement the peer protocols. Often these local protocols may be viewed as occurring "vertically" between software modules at different protocol levels in the same node (for example, between session layer and transport layer modules in the same node), in contrast to the peer protocols which are often view as occurring "horizontally" between protocol modules at the same level in different nodes. However, sometimes local protocols are also needed between peer protocol modules at the same level in the same node (for example, between a file transfer protocol module and a name server protocol module in the same node). The local protocols are required for a variety of purposes, and the implementation has to take account of many additional matters not considered in the specification for the peer protocols, including:

a. The management of memory buffers to handle protocol messages

b. The management of tasks to control the handling of protocol messages

c. The management by tasks of intertask interaction sequences to achieve correct system operation

d. The passing of control and data back and forth between the modules implementing the different peer protocol levels

These matters are in the domain of the local protocols.

The local protocols are implemented via the control structures of the software execution environment and are often particular to that environment. As such, they are traditionally viewed as part of the software design process rather than the protocol design process.

However, there may be advantages from a combined environment viewpoint in treating them formally as protocols because this could lead to a greater degree of automation in the software design process. One purpose of the baselining activity is to determine whether or not this is desirable.

For WIS purposes, the peer protocols are those required for the wide area network (WAN) and the local protocols are between software modules in the hosts of the LAN. Because the WIS LAN OS is planned to provide transparency of the LAN to Ada programs, it does not matter whether the modules communicating via the local protocols are on a single host or on different hosts of the same LAN. Therefore, the local protocols will be considered to be between software modules on a single computer, understanding that this includes the WIS LAN case.

3.2.3 Issues

This section identifies issues relative to automatic code generation from both a protocols perspective and a design environment perspective.

3.2.3.1 The Issues from a Protocols Perspective

3.2.3.1.1 Specification Technique

What specification technique should be used for the peer protocols? Techniques in current use include narrative text combined with timing diagrams (this is the old way, which has caused problems in the past), coupled state machines, petri nets, temporal logic, and temporal sequencing description languages. Of the formal techniques, the state-machine-based ones are perhaps the best known and most widely used.

3.2.3.1.2 Software Design Environment

What techniques should be used to arrive at a suitable software design (using this term here to denote the control and module organization of the software) for a particular set of protocols? Currently this is a matter for human design judgement. Can and should the process be automated and to what extent? The answer depends on the flexibility needed for choice of software design paradigms. The greater the flexibility required, the greater the amount of human input required. If the human designer is to have complete freedom of choice of software design paradigms, then human input of software design decisions must be supported. A more automated system might provide the designer with a choice of paradigms, within which standard software organizations are employed. A simple approach to a fully automated system would be to base it on a single software paradigm. However, given the wide range of possible paradigms possible with Ada, embracing both non-tasking and multi-tasking organizations, it seems difficult to choose a single suitable paradigm.

3.2.3.1.3 Local Protocols

How and at what stage of the process should the local protocols be specified? Because these are between modules of the software in a particular computer, some decisions must be made about the software design first. Are there levels of specification in the local

protocols some of which should affect this design and others of which can be left until after it is decided? Should the local protocols be combined in part with the peer protocols in a compound protocol specification before proceeding with software design or code generation? Should the local protocols never be explicitly specified as protocols at all, but only enter the system as aspects of the software design? In current practice, these are matters which are left up to human designers and implementors; there are no widely accepted methods in use.

3.2.3.1.4 Modularity

How should modularity be introduced into the specification and design process? A conflict exists in this area between the requirements of the peer protocol specifiers, who do not want to make implementation commitments, and those requirements of the software designers whose work may be helped by the presence of modularity in the protocol specification. On the one hand, it is difficult to introduce modularity into the protocol specifications without implicitly making decisions about implementation organization. For example, some Estelle specifications can have many or few modules and channels, depending on the specifier's choice; the particular choice may constrain the implementation organization. On the other hand, protocol specifications without such modularity are likely not only to be complex, but also to be difficult to translate into modular implementations in a straightforward manner. A further issue associated with modularity is that of races. One way of achieving modularity in a protocol specification is by specifying the whole in a distributed manner as a set of coupled protocol machines; such a distributed specification may inadvertently introduce races. Is modularity worth this risk? If so, how can the risk be controlled?

3.2.3.1.5 Combined Environment

Assuming the environment allows the user to control the software design, how can the mechanisms for protocol specification and software design be combined so that both activities may be pursued independently in any order and the results later integrated? Required is a compatible input mechanism, with support for performing the integration.

3.2.3.1.6 Validation

How can the combined environment contribute to the validation of the results? Although techniques are emerging for validating specifications for peer protocols, these techniques do not necessarily validate the resulting software designs or implementations. Indeed, it is conceivable that different implementations of the same protocol specification could be incompatible with each other, due to a combination of incompleteness of the specifications and different implementation decisions. The combined environment can help through the automatic generation of test sequences for the protocols and of test software to be included in the implementation. How is this to be accomplished?

3.2.3.2 The Issues from a Software Design Environment Perspective

The environment must support the following aspects of the software design process through appropriate interfaces, databases and tools:

a. Structural Information: This defines the objects in the system (e.g., Ada packages and tasks) and the paths of interaction between the objects (e.g., calls, propagation of exceptions). Data flows and the structure of interface parameters are included.

b. Temporal Information: This defines sequencing relationships between events in the system. Linkages are identified which specify the order of events and the structural objects involved.

c. Action Information: This defines specific actions to be taken in response to events. The actions are triggered by events and usually involve data manipulation and the propagation of events (creating new events in response to the trigger event).

Particular issues in these areas are identified in the following sections.

3.2.3.2.1 Graphics

Should graphics be used as the primary input mechanism for structural, temporal and action description? Should a language approach be used? Or is it better to have compatible language and graphical input mechanisms, either of which may be used?

3.2.3.2.2 Ada for Protocols

How best can the "advanced" features of Ada such as packaging, tasking, exceptions and generics be catered to by the methods for specifying structure, temporal and action information?

What specific features of Ada are particularly useful for protocol systems? Possibilities include using Ada packages or tasks for state machines, using task entries for protocol waiting conditions, and using timeouts on calls and accepts of task entries for protocol timeout conditions.

3.2.3.2.3 Compatible Protocol and Software Design Specifications

How can we specify interfaces (intermodule data flows, interaction queues, stimulus events, etc.) in a manner compatible with both components of protocol specifications and components of software designs?

How can protocol control requirements such as timeouts and waiting conditions be defined in a manner compatible with the definition of suitable software control structures for implementing them (e.g., in Ada the possible use of task entries for waiting conditions and timeouts)?

How can data structures and the actions on them be described in a manner compatible with protocol specification, software design and code generation?

How can the problem of incompatible modularizations in the protocol specifications and the software designs be avoided or resolved?

Should the software design be specified at a higher level than that of Ada objects, with the Ada objects required by the design being automatically generated by the environment according to some predefined paradigm?

3.2.3.2.4 Temporal and Action Descriptions

What are the relative advantages of using state-machine-based methods versus other methods for specifying temporal and action information?

3.2.3.2.5 Structure-Based Versus Temporal-Based Approaches

Are there advantages to taking a structure-based or temporal-based approach? Or should the environment provide a choice of compatible approaches? A structure-based approach is one in which software control structure is defined first and the temporal and action information added later associated with components of the structure. A temporal-based approach is one in which the software structures are subservient to the temporal and action information. The approaches are not necessarily incompatible, but experimental environments seem to be noticeably biased towards one or the other.

3.2.3.2.6 Environment Implementation

What are the advantages or disadvantages of implementing the environment in Ada versus some other approach with greater flexibility (e.g., Expert System Shell such as [KEE 84], Prolog, Lisp, Smalltalk)? In a sense, this is a non-issue for this project, because of the mandated use of Ada for it. However, the baselining activity should identify any difficulties this may present.

How should the environment software be organized? There may be modularity advantages to organizing it as a rule-based system following an expert system approach. Should this be done? How can this be done in Ada?

3.2.4 Review of Existing Work

In this section, highlights of the relevant existing work are reviewed in the areas of formal specification techniques for protocols, automatic code generation for protocols, and software design environments for embedded systems. This section does not attempt to be comprehensive survey of the literature. It is anticipated that the implementors of the baselining projects will need to fill in the gaps and in some cases make direct contact with key workers and projects to get the latest information. The concern of this section is to identify key approaches, workers and projects. Accordingly, references are sometimes made just to known names and projects rather than to specific publications. Occasionally references are made to verbal reports.

3.2.4.1 Existing Work in Formal Specification Techniques for Protocols

Research work on formal techniques for protocols is concerned with peer protocols and has mainly been motivated by the following needs:

a. To validate protocol specifications for correctness, independent of their implementations.

b. To describe protocols precisely, to guide the (human) implementors.

c. To generate implementation code automatically.

Not all of this work addresses the third need.

3.2.4.1.1 State-Machine Techniques

Two state-machine-based techniques have achieved a substantial measure of international acceptance to date. These are the ISO's Estelle and the CCITT's SDL, both of which are described below. Both of these techniques have been used as the basis for automatic code generation experiments in the research laboratory.

Estelle (standing for Extended State Machine Language) [ISO 84A] is a language developed by the ISO which is likely to emerge soon as an international ISO standard. It is an extension of Pascal. It provides for declaration of state-machine modules, declaration of interaction channels and service primitives between the modules, and specification of rules for the transitions and actions of the state machines. It does not define implementation mechanisms in standard Pascal for the interaction channels and primitives. Thus an Estelle program is not executable. The modules and channels are only intended as means for partitioning the protocol specification (for a peer protocol) into manageable parts. They are not intended to imply anything about the organization of implementations. However, in practice it turns out to be difficult to avoid making decisions about the organization of implementations when writing an Estelle specification, because of the difficulty of mapping an Estelle specification with a particular module organization onto an implementation organization with a different one.

SDL [CCITT 84B] is a graphical notation for specifying interacting state-machines to implement protocols. It has aspects both conventional state transition diagrams and of flow charts for describing control flow in computer programs. It is an international CCITT standard. It was originally developed to describe low level interactions with subscriber telephones in switching systems and still conveys that flavor. However, it has been greatly extended. Like Estelle, it forces the protocol specifier to make decisions which implicitly affect the organization of the implementation. The CCITT is reported [MCRU 85] to find SDL completely adequate for its current needs and to be uninterested in the adoption of Estelle as a standard or in the alignment of Estelle with SDL as a standard.

3.2.4.1.2 Petri Nets

Telecom Australia has developed the use of Numerical Petri Nets for protocol specification. [CCITT 84A] This work has been used as the basis for an automatic code generation experiments. However, in spite of strong efforts in the international CCITT arena by Telecom Australia, this approach has not been accepted for international standardization by the CCITT. [MCRU 85]

3.2.4.1.3 Temporal Languages and Logic

The ISO is developing a temporal ordering language called LOTOS. [CCITT 84A, ISO 84B] LOTOS is different from Estelle in two ways: it explicitly represents concurrency and it is executable. This language is also likely to become an ISO standard, along with Estelle.

Temporal logic [RESC 71] has been explored by a number of workers [HAIL 82, CAVA 84, WOLP] as a means of specifying protocols and has been proposed by at least one worker as a basis for automatic code generation. [WOLP]

3.2.4.1.4 Other Approaches

Zave has proposed a sequence diagram approach based on the Jackson software design methodology which is formally equivalent to state-machine approaches but which she claims has better modularity properties. [ZAVE 85] The specifications are supposedly not only to be easier for humans to understand, but also to be free from the race-proneness that is a danger when modularizing state-machine specifications.

Logrippo and others at the University of Ottawa have used Prolog for executable specifications of protocols. [LOGR 84] Koomen [KOOM 85] uses an algebraic method for specifying and verifying communication protocols.

3.2.4.1.5 Testing

The ISO is working on a methodology for protocol testing [ISO 85] which will be helpful in the problem of the automatic generation of test sequences and test software. Executable Prolog specifications have been used for test sequence generation by Logrippo and others. [LOGR 84]

3.2.4.1.6 Future International Standards

The adoption of a universally accepted formal protocol specification technique remains in the future. The CCITT is reported [MCRU 85] to be interested in evaluating candidates for "the protocol specification language of the future," which it believes will be neither SDL or Estelle, but something different. What shape this language is likely to take is at present unknown.

3.2.4.2 Existing Work in Automatic Code Generation for Protocols

3.2.4.2.1 State-Machine Methods

Both language and graphical forms have been used for the specifications. Automatic code generation may be performed by translating individual state-machine specifications into program fragments (usually subprograms) in languages such as Pascal or C. However these subprograms do not by themselves constitute complete programs. Human intervention is required to devise a control framework appropriate for a particular hardware and operating system environment and to integrate the automatically generated procedures into this framework. Further human intervention is required to validate the results.

With the code fragment approach to automatic code generation, the work of inserting the fragments into an appropriate control framework can vary from being quite simple to quite complex. The simple case, which seems ripe for earliest automation, is for protocols which are of the initiator/responder type in which control passes back and forth between the initiator and the responder in a highly predictable fashion. Little concurrency is required in the protocol software for such cases and the control structure mainly has to ensure that procedure calls occur in the correct sequence. However, even in this simple case, some concurrency is required because of the need to handle unexpected events such as aborts. More complex protocols require more implementation concurrency.

Bochmann [BOCH 85] at the University of Montreal has implemented a translator which produces executable Pascal procedures from protocol specifications written in Estelle. (Bochmann was also a prime mover behind the development of Estelle.) Human intervention is required to provide the calling framework in Pascal for these procedures.

The National Bureau of Standards (NBS) implemented a translator for an Estelle-like protocol specification language of their own invention. [NBS] It produced C code. Hand tailoring was required to add the control framework. Arising out of the NBS work, the Protocol Development Corporation is working on an Estelle development environment product. [PDC 85]

A student at Ottawa University implemented an Estelle to C translator using Unix's Lex and Yacc tools. [PROB 85]

SDL [CCITT 84B] has been the basis for a number of systems and experiments with automatic code generation in Britain, Europe and Australia. One example is British Telecom's CADOS system [CADO 85], which translates SDL graphical input into code fragments in a variety of languages. Another is the Australian MELBA system [FIDG 84], about which more will be said later. Work has also been performed at various places in Europe, as reported in the proceedings of the Fifth International Conference on Software Engineering for Telecommunications Systems. [SOFT 83]

Harry Rudin is understood to have been responsible for work at the IBM Zurich research laboratory on a translator from protocol state-machines to software, which was later applied to IBM's SNA protocol at the San Jose research laboratory [RUDI]. At the time of this writing, no additional information is available.

3.2.4.2.2 Petri Nets

Telecom Australia's work with Numerical Petri Nets and SDL is reported to have produced a code generator called Protean. [PARK 85] The code generation sequence is understood to involve first going from SDL to Numerical Petri Nets. Whether this is manual or automatic is not known at the time of this writing. Computational effort required is understood to be relatively high (a verbal report indicated a Vax 780 for a weekend for an unknown version of the ISO's transport protocol). The software is understood to be written in Pascal. The target language is either Pascal or CHILL.

3.2.4.2.3 Temporal Languages and Logic

Both LOTOS and Temporal Logic have been proposed as the basis for automatic code generation, but specific examples of experimental work are not known at the time of this writing.

3.2.4.2.4 Experience

Verbal reports have indicated that some automatic code generation work encountered difficulties with the large size of the code produced, to the extent that it required substantial hand tailoring to cut down its size. There are also verbal reports that some of the code generators are very large and demanding of processor time. Pinning down such problems for Ada is one of the purposes of the baselining activity.

3.2.4.3 Existing Work in Design Environments With Particular Emphasis on Embedded Systems

The combination of a concurrent protocol with a concurrent implementation framework opens up design problems currently in the domain of experts in both protocols and concurrent software design. These design problems are familiar ones in the embedded system area. Accordingly, work in design environments for embedded systems is relevant for this project. There is a lot of international activity in this area currently.

3.2.4.3.1 Design Environments Targeted Specifically for Embedded Protocol Systems

Two projects are known to be tackling head-on the problem of design environments for embedded protocol systems. These are the MELBA project in Australia [FIDG 84]] which started in 1979, and the CAEDE project in Canada [BUHR 84-85], which started in 1982. These two projects provide vectors for this project.

MELBA and CAEDE have substantial similarities in both philosophy and approach. Both have iconic front ends for specifying system structure in terms of black boxes and their interconnections. Each bases its design metaphors on a concurrent high level language: CHILL in MELBA and Ada in CAEDE. Both provide some support for automatic code generation. Each allows the designer great freedom in specifying the software organization. MELBA uses SDL for describing the temporal behaviour of its modules. CAEDE is currently developing its own graphical approach, but could in principle use SDL. Both MELBA and CAEDE aim at developing complete designs embracing structure, temporal behaviour and action, from which relatively complete code can be generated. While MELBA is further advanced than CAEDE in the automatic generation of code bodies from design input, including both abstract data types and temporal behaviour, CAEDE has several unique features relative to the testing of preliminary designs before code generation.

3.2.4.3.2 Flow Graph Based Approaches

The Very High Speed Integrated Circuits (VHSIC) program has produced a number of signal flow graph based approaches targeted at software/hardware codesign which may contain useful ideas for this project.

A directed graph methodology has been developed [VHSIC 83] which allows the designer to specify a system in terms of a signal flow graph whose nodes may be either subgraphs or primitive program components. A fixed software organization paradigm is used for the primitive program components: nodes are implemented as Ada tasks, with associated buffer tasks for input and output along the arcs of the graph. There may also be master control tasks. Code may be automatically generated from the graph input following this paradigm.

This paradigm is a natural one for signal processing systems because it is common practice for designers of such systems to use signal flow graphs as the starting point for design. The design approach and the underlying software paradigm may be too restrictive for protocol systems in general, but the idea of a simple, application-oriented front end providing input which can be used to generate code according to a suitable paradigm is very appealing.

Another VHSIC-oriented project following a similar approach is described in [SMITH 85].

3.2.4.3.3 Other Graphics-Based Approaches

The Mascot system [MASC 80] is an example of a graphically oriented design method for embedded systems.

A project is believed to be underway at the Mitre Corporation in Bedford [MONK 84] to use SADT as the front end for an embedded system software generator which will be capable of generating concurrent software. This approach is not viewed at this time as an appropriate one for this project because of well known difficulties in automatically

translating SADT designs into programs. [ROSS 85] However, if Monk succeeds in overcoming these difficulties, the results will be worth looking at.

3.2.4.3.4 Petri Net Based Approaches

Petri nets have been used for concurrent software design and description as well as for protocols.

Petri nets have been used by Cherry [CHER 84] as a means of describing the temporal logic of concurrent Ada programs before programming them.

Predicate transition nets (a form of Petri nets) have been used by Kramer [KRAM] as the basis for a language approach to stepwise construction of non-sequential software systems and by Bondeli as a basis for describing the temporal behaviour of Ada programs. [BOND 85]

Kramer makes a point that is sometimes not fully appreciated by advocates of net based approaches. He says: "A weakness of conventional net models ... is that concepts for abstraction and for structuring systems-in-the-large (i.e., for stepwise implementation, modularization and scoping) are underdeveloped."

3.2.4.3.5 Temporal Language and Logic Based Work

Several current projects are tackling the problem of temporal behaviour specification for real-time systems in a manner which could offer guidance for this project.

Luckham's group at Stanford is developing a language called TSL to describe temporal sequencing in Ada programs. [LUCK 85A] Although the original aim of this work was to describe output sequences for debugging purposes, TSL has also been applied as a design tool for a protocol implementation experiment. [LUCK 85B]

Temporal logic has been employed by Wolper at Stanford as a basis for synthesizing concurrent programs. [WOLP] He argues that his approach provides a useful approach for generating protocol software.

Other relevant work in the real-time systems area includes that of Balzer at ISI [BALZ], Alford [ALFO 77, 84] and Ellis [ELLI 83] at TRW Huntsville , Kok at the University of Texas at Austin [KOK 84], and several workers at the University of Massachusetts. [CLAR 84]

3.2.4.4 General Work

Frenkel [FREN 85] provides an overview of work on automating the software development cycle. Key points which emerge from this overview areas follows:

a. The importance of modularizing the code generator by separating mechanisms from rules which guide choices and optimizations.

b. The importance of incorporating human software expertise in the rules which guide the code generation.

c. The usefulness of expert system techniques in modularizing the problem.

In an interesting interview report [ROSS 85], Doug Ross, the inventor of SADT, in recounting his experiences (apparently not particularly successful) with automatic code generation from ADT, says about the possiblity of developing a good tool for transforming SADT to code: "... my guess is that the analysis will interface not to a generator of working code, but to software building blocks of considerable size." Ross seems somewhat pessimistic about the near term possibilities of generating good code automatically from SADT and we do not regard SADT as a promising candidate for the front end of this project.

However, Ross's remark has wider significance. The design environment which is our ultimate target will need to optimize the organization of the software in terms of building blocks at the highest possible level. The need to tinker with generated software at too low a level will likely be an admission of failure.

The project may be able to take advantage of Luckham's ANNA language [LUCK 84] to specify protocol actions.

3.2.5 Particular Problems Associated With DoD Protocols

The major problem associated with DoD protocols is that the ISO and CCITT work in formal methods has not used them as examples. The only example known to the authors of this document of a formal specification of the DoD protocols is a project under Agrawala at the University of Maryland. [AGRA 85] However, this was not done using one of the formal methods accepted by the international ISO and CCITT communities.

3.3 Approach

A provisional list is provided in Section 3.3.1 of general requirements for the 1990's combined environment. This list provides criteria for the two baselining projects. The detailed approach to these two projects is then described in Sections 3.3.2 and 3.3.3.

3.3.1 Requirements for the 1990's Combined Environment

A provisional list of general requirements for the 1990's combined environment is given below, to be used as a guide. It is expected that these requirements will be refined and extended by the baselining projects.

a. There must be input capability not only for protocol specifications and program design specifications, but also for rules of various kinds affecting the nature of the code to be generated. The kinds of rules required are enumerated later. However a key point is that, for modularity, the rules must be enterable independently of the implementation details of the code generator.

b. The environment should not force a single software organization paradigm on the user, but rather should provide freedom of choice of paradigms within limits.

c. The programs generated should be neither significantly larger nor significantly slower than hand-generated Ada programs for the same purpose.

d. The Ada programs generated should be well structured at all levels of abstraction, to provide not only for human monitoring and evaluation during prototype development, but also because this will be a natural byproduct of the requirement for optimization-in-the-large.

e. The environment should be capable of being integrated into a 1990's standard Ada software development environment. (Note that in accordance with current policies this means it must be implemented in Ada.)

f. The environment should make maximum appropriate use of graphics as an input and display medium in order to make the human interface as powerful as possible. The graphics should be integral to the design method and not simply used for display.

g. The environment should be capable of dealing with the specification of programs and protocols at many levels of abstraction including Ada code, high-level descriptions of data structure updating, still higher level descriptions of temporal sequencing of protocol actions, and, at the highest level, descriptions of the patterns of organization of programs and suites of protocols.

h. The environment should support a dynamically updateable library of standard protocol system building blocks at all levels of abstraction.

i. The environment should not require the user to enter Ada program statements to specify any part of the protocols or software modules. However, it should be possible to include blocks of canned Ada code in the generated implementation.

j. The environment should support modular specification of protocol rules and software organization. It should be possible to specify protocol rules and software organization separately and to perform the integration later or to specify both jointly. It should be possible to change one without affecting the other, providing the changes do not affect interfaces.

k. The environnent should be capable of accepting such specifications in standard form to be used as the starting point for the software design because peer protocol specifications may be done well in advance of software design. It should be possible to enter suites of peer protocols in this fashion.

l. The environment should be capable of accepting additional local protocol specifications in standard form as augmentations of peer protocol suites and of integrating the whole into an appropriate software organization, guided by both designer input on the nature of the software organization required and internal rules.

m. The environment should be usable by protocol specifiers who are not expert Ada programmers.

n. The environment should be usable by Ada program designers who are not protocol experts.

o. The environment should be capable of dealing with all intended WIS protocols above the LAN level. (WIS OS is regarded as providing a distributed run-time system for Ada in the LAN, so that the entire LAN is regarded as a single node from the WAN protocol point of view.)

p. In addition to code generation, the environment should support automatic generation both of test sequences and of test software.

q. The environment should be capable of supporting tools for analysis and validation of incomplete protocol software designs. It should not rely solely on execution of generated code for validation.

r. The environment should be capable of using its rules for generation of program organizations and for criticism of program organizations entered by human designers.

Outside the scope of the environment is the analysis of the protocol specifications themselves for correctness and quality. The assumption is that current research in the protocol field will produce methods for doing this on protocols specified in standard forms. The challenge here is to implement these protocol specifications as correct, well-structured, compact, efficient programs, using the maximum possible degree of automation.

For modularity, it is essential that the environment be rule-based with a capability to enter the different sets of rules in an independent manner and without knowledge of the code generator implementation details. This is not meant to imply that the environment must be implemented as an expert system, although this is an obvious approach. It is expected that the baselining projects will refine and expand the following list of examples of the types of that will be required:

a. Program organization rules at all levels of abstraction, covering program quality, compactness and efficiency

b. Protocol temporal behaviour rules (e.g., via state machines)

c. Rules for generating code from protocol temporal behaviour specifications

d. Rules for inserting standard program interfaces between protocol modules, to enable completion of incompletely defined interfaces, or criticism of badly structured ones

e. Rules for determining when "boiler plate" code is required for standard purposes such as queueing and for inserting the appropriate code and associated interface definitions

Such rules provide the basis for generating code, for optimizing its organization and for criticizing organizations entered by human designers. The success of the project will depend on the ability to find optimizing rules for program organization which can operate at a high level of abstraction in such a way that their will be little need for optimization at lower levels. This may be called optimization-in-the-large. Particularly to be avoided is the need for analysis of generated Ada programs on a statement-by-statement basis to perform optimizations.

It is left for further consideration whether or not the mandated use of Ada for the 1990's environment requires the rules themselves to be expressed in Ada, or only the rule interpreters.

3.3.2 Approach to Attaining the Short Term Objectives

This description of the approach to satisfy the short term objectives assume a two-pronged attack conducted in parallel. Other than making interim reports available for information, there is no specific coordination required between the contractors for the parallel projects.

3.3.2.1 Approach to First Short Term Objective (Code-Fragment Generator)

As stated earlier, the first short term objective is to be pursued by repeating the intent of selected experiments in automatic code fragment generation performed in the protocols community for other target languages than Ada, taking due account of both similarities and differences between Ada and the other languages. The emphasis is on learning how to generate the best possible Ada target code, taking advantage of unique features of Ada such as packaging, tasking, generics and exceptions. Criteria of goodness for the target code are included in the requirements defined earlier for the combined environment.

The experimental work for the first short term objective is to be approached using a rapid prototyping, rule-based approach, consistent with the requirements for the 1990's combined environment given above. For the preliminary experimental work, it is recommended that the implementor make as much use as possible of existing rapid prototyping environments and tools. It is anticipated that environments with their roots in artificial intelligence (AI) techniques and languages may offer advantages for the experiments.

In accordance with DoD policy, the final code generators must be in Ada. However, it is not the aim of this project to produce code generators in Ada which will be reusable in the combined environment of the 1990's.

The particular existing work which is to be extended to Ada in this project is the following:

a. Experiments with generating Pascal and C code from Estelle specifications, following in particular the work by [BOCH 85], [NBS] and the Protocol Development Corporation.

b. Experiments with generating code in a variety of languages from SDL specifications (language form of SDL), following the work of selected key members of the CCITT community. (A starting point is the CADOS work in the UK and similar work in Australia and Europe.)

A single protocol is to be selected and experiments conducted in generating code for this protocol using the different methods. For this purpose, a suitable, well-studied protocol is the ISO transport protocol, Class 4.

In extending this work for Ada, the implementor should pay due attention to gaining insight into the issues and requirements outlined in Sections 3.2 and 3.3 and to report this insight in the form of discussion and recommendations.

The intent in this work is not to build a prototype design environment capable of producing complete Ada programs. Issues in building such an environment will be investigated in this project by experimenting with manual creation of complete programs from the automatically generated fragments.

Based on the insight gained in performing the experimental work, the possible use of other protocol specification methods as the basis for code generation should be reviewed and evaluated. The report is to include a complete, annotated bibliography on relevant work.

3.3.2.2 Approach to Second Short Term Objective (Combined Environment Study)

As stated earlier, the second short term objective is to be pursued as a paper study, taking account of the literature and of direct contact as necessary with selected researchers and laboratories to get current information.

The first step is to itemize the requirements for the 1990's environment which will be used as the basis for all the analysis. The requirements provided in Section 3.3.1 are intended to serve as a starting point, but may require elaboration.

The next step is to provide a detailed characterization of the approaches to be evaluated. Broadly speaking, these approaches fall into the following classes:

Class I Egalitarian Environments

Egalitarian environments are ones in which the specification of software designs and the specification of protocols both have equal status. The problem is to find a way of specifying the two in a compatible manner and then of integrating them. This class of environments has the advantage of being adaptable for use by either software designers or protocol specifiers. Current work on CAEDE is aimed at producing such an environment. MELBA may already be one.

Class II Behavior-Driven Environments

Behavior-driven environments are ones in which the software organization follows from a description of the desired temporal and functional behavior, according to predefined software organization paradigms. The environment user is given less freedom in specifying the software organization. A possibility for this class of environments is to allow the human user to specify rules to define the software paradigms.

This class of environments is inherently application oriented (in this case, protocol oriented). A distinguishing feature from Class I is the inherent requirement that the environment contain substantially more application-oriented intelligence in order to generate appropriate software automatically.

The Estelle environment being developed by the Protocol Development Corporation may be an example of this class, for a restricted software paradigm. The VHSIC signal flow graph work is an example of this class for a different application area, that of signal processing. Alford's work is an example of this class in the real-time system area. [ALFO 77]

Class III Software-Driven Environments

Software-driven environments are ones in which the user must specify the software organization first and then the internal details of the software modules to implement the required protocols. The current version of CAEDE is an example of such an environment.

Within these classes, many different possibilities exist and must be explored for the nature of input, processing and output. The combination of a class of environment and particular methods for input, processing, and output is said here to constitute an "approach".

The analysis should proceed by evaluating the set of defined aproaches against the requirements, identifying strengths and weaknesses of key aspects of the approaches.

A recommended method is to analyze how the different approaches might be applied to the specification and design of software for a representative set of protocols. A suitable set is the ISO's transport protocol, Class 0, and the Session Kernel Functional Unit.

4.0 DEVELOPMENT AND EVALUATION OF MULTIVARIABLE OBJECTIVE FUNCTION NETWORK ROUTING FOR WIS

4.1 Introduction

4.1.1 WIS Requirements

WIS is composed of a moderate number (less than 50) local area networks (LAN's). They are located worldwide but are grouped at a number of sites, e.g.,Washington, D.C. The LAN's service a heterogeneous mix of terminals and hosts. The Defense Data Network (DDN) will be used as the long haul backbone. Interoperability is required with other networks, domestic and foreign, commercial and military. WIS requirements that are unique and/or stringent include multi-level security, priority, reliability, reconfigurability, and distributed databases. Applications to be serviced include teleconferencing, Telnet, file/data transfer, wide-band voice and data, narrow band voice, video, and facsimile.

The objective of the work outlined in this specification is to develop and evaluate approaches to routing in the World Wide Military Command and Control System (WWMCCS) Information System (WIS) network. The work will result in the development of prototype mathematical software and will specify mathematical models with which to find routes that optimize multiple objectives subject to given constraints such as bandwidth and delay. A service dependent routing is required where different applications needing different services are routed using criteria suited to that application. Routing algorithms will be developed and evaluated with respect to efficiency and performance. The main effort of this work will be to develop and evaluate these algorithms. During the course of the work, the impact of the routing algorithms on network protocols will be examined and a methodology defined for gathering the relevant data and to report it to gateways for routing and to the data sources for use in evaluating the performance of applications. In addition, concern should be directed to how the routing algorithms are implemented in the gateways, to gateway-to-gateway protocols, and to the implementation of the gateways with respect to handling priorities and other tasks dictated by the routing.

4.1.2 General Assumptions

The DDN will be based on a datagram service similar to, if not identical with, the current DARPA set of protocols. The nature of the LAN protocols is not clear at this time. Nearby LAN's will not be interconnected directly. Thus all LAN's will be connected via multiple gateways to the DDN. The DDN will provide gateways to non-WIS networks. The LAN to DDN gateways will be responsible for routing, other network services, and providing additional services required by applications of WIS that are not currently provided by standard internet protocols. Most communication between hosts will be over virtual circuits. The remainder will be in the form of datagrams.

4.1.3 Architecture

There are many LAN's interconnected by DDN. Each LAN is connected to DDN through two or more gateways for the sake of reliability. Each of the gateways for a LAN is connected to one or more different portions of DDN. Consequently a single failure in DDN cannot remove a LAN. LAN's will not be connected via bridges. The DDN will provide connections between all gateways. Its own routing procedures will determine best paths. Criteria used by DDN will be its own, e.g., shortest hop or minimum delay, or selected

from a menu by WIS. Different criteria could be used by different classes of traffic. It is unclear at this point in time what information will be made available by DDN for use in routing decisions. One possibility is that DDN reports to gateways the quality and grade of service of paths. Alternatively, WIS gateways would probe the DDN to derive this information.

There will be several paths between hosts on different LAN's. First the host will select the gateway to be used to leave its LAN. The decision could be made at the initiation of the virtual call by accessing a name server and router on the LAN. The selection of source gateway could be changed in the middle of a call. Provisions would be needed for determining when this is necessary and informing the host to change its routing. The gateway could return a message to the host who would re-interrogate the router. The remainder of the routing will be on a datagram basis. The source gateway will determine which destination gateway to use for the destination LAN and which DDN access to use. The entire route selection, gateways and DDN access could be done by a single router on the source LAN. The choice of route will be based on class of traffic, priority, and grade of service required. The source LAN will concern itself with its own loading, primarily in the gateways, path quality information from the DDN, and loading and other information from destination gateways.

In the remainder of this document, it will be useful to focus on the following two figures. The first figure (Figure 6) illustrates communication between a source on one LAN and a receiver on a different LAN where each LAN is connected to DDN by two gateways.

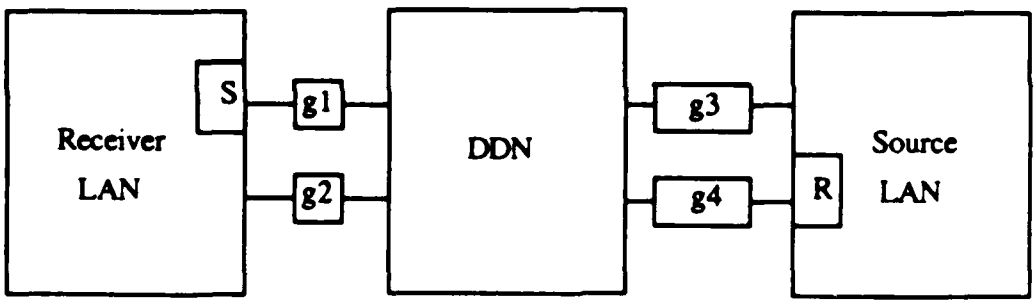


Figure 6. Inter-LAN Communications

The second figure (Figure 7) illustrates the logical structure of a LAN with regards to routing. There exist users (U1, ..., Um) with data to transmit to other users, name servers (N1, ..., Nk) to obtain physical addresses, routers (R1, ..., Rn) to determine routes to other LAN's, and gateway functions

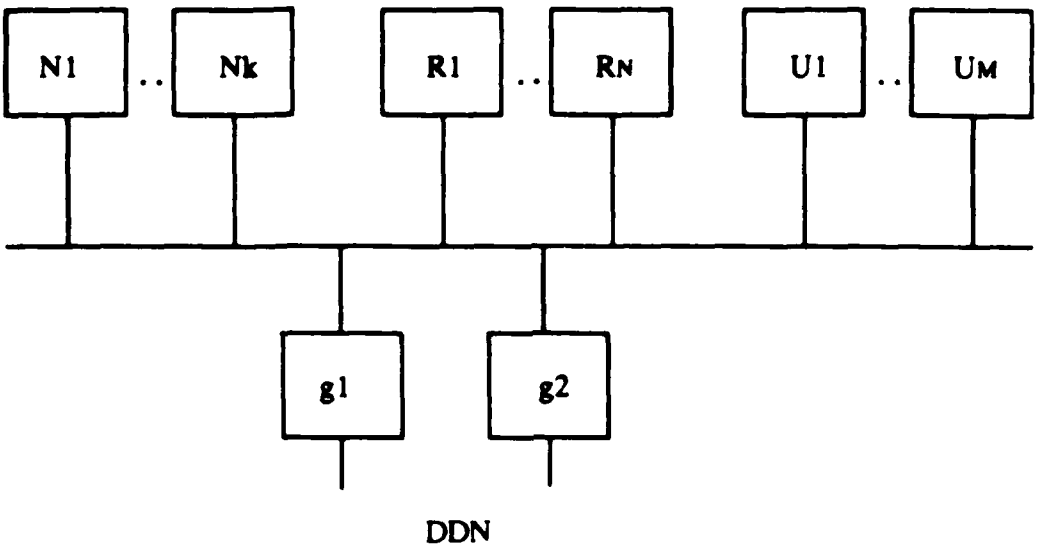


Figure 7. LAN Routing

Physically, there may be any number of name servers and routers and they may reside anywhere, including the gateways.

4.1.4 Protocol Issues in WIS

There are several aspects of inter-LAN communications that need to be addressed before an inter-LAN mechanism can be implemented. First a good routing protocol must be developed. Second the gateway architecture must be specified. The routing protocol itself consists of an algorithm for computing the routes based on some cost functions and the actual protocol that uses the route information to perform the inter-LAN communications. These last two shall be referred to as the routing algorithm and routing protocol. The main concern of this document will be the routing algorithm itself. These algorithms depend upon and must be consistent with protocol issues. A secondary concern for the first phase is to explore this consistency. Some protocol and additional architectural issues are described below.

There are a number of issues that have to be addressed before efficient, reliable inter-LAN communications can be achieved, including a concern with optimal routing for systems having multiple objectives. As stated earlier, numerous kinds of applications will be supported by WIS. Some applications must satisfy real-time constraints. Other applications will require a minimum bandwidth. Yet others will require a high reliability. In addition, applications may be assigned different priorities which may or may not have any correlation to the kind of application that it is. It is unclear at this point what the interplay between kind of service and priority is and how it will affect the routing algorithms and protocols.

In addition to handling multiple objectives and priorities, the routing algorithms and associated protocols should be capable of adapting to changes in the traffic load and the topology of WIS. Two kinds of network changes are envisioned. The first corresponds to slow changes as might occur during normal operation. The second corresponds to the

advent of a crisis where the network topology may change dramatically within minutes, and subsequently, the demands placed on WIS change dramatically.

A second issue is that of the kinds of routing services that should be available to WIS. Besides the standard single destination routing, there will also be a need for multiple destination routing. An application may be served by resources at one of several locations. The choice of location depends upon the availability of the network, hence the routing, as well as the loading of the resources. At this point, it is unclear how widely this facility will be used or who will control the process, the LAN operating system or the router? There will also be broadcast messages to a set of destinations. Of concern is how to avoid unnecessary duplication. Is this a service available from, or to be requested from DDN? How are acknowledgments handled?

A third issue arises from the requirement that WIS provide the capability to applications to determine whether they can be served efficiently or not. A response to this type of question would allow them to determine how they should configure and/or whether they should even execute. If an application requires resources at more than one LAN, then it will require information from the inter-LAN protocols regarding the expected performance available during inter-LAN communications. Consequently, information used by the routing algorithms may be useful to the higher level protocols.

A fourth issue relates to how virtual circuits should be treated by the routing protocols. Specifically, should virtual circuits be assigned fixed source and destination gateways? Should they be assigned to specific source gateways but allowed to use either destination gateway? The last alternative of course is to allow a virtual circuit to use any gateway. If either of the first two approaches are used, then how will the routing algorithm handle drastic changes in either the topology and/or traffic load?

Protocol issues for routing in packet radio networks are treated in [KAHN 78], [MACG 82] and [WEST 82].

4.2 Routing Algorithm Issues

This section discusses some of the issues specific to routing algorithms that require solution in any network setting, including the WIS system.

The designer will be concerned with developing routing strategies that are concerned with optimizing different cost functions (e.g., average delays, number of packets satisfying real-time deadlines, probability of failure), different kinds of traffic (e.g., voice, files, electronic mail, procedure calls, etc.), different priorities, etc. Different applications require different services. The routing algorithms should satisfy criteria suitable for each application.

Routing in WIS and in DDN will be done co-operatively. Some functions will be performed by each, others will be shared. Information will have to be exchanged to implement the routings, select from options, and report performance. Solutions to the routing and protocol issues addressed in this work will be shared with the DDN so that they can form the basis for additional services that DDN will provide and address other issues of concern to DDN and its users.

The routing protocol should provide good performance in an environment of changing traffic workloads. Moreover, this protocol should be able to adapt to drastic changes in the structure of DDN and the gateways. It should be able to identify a path whenever it exists between two LAN's. Over time it should attempt to develop routes that yield good performance within the new topology. The following problems arise. What kind of

dynamic routing should be present at the gateway? What kind of metrics are useful? How should the metrics be determined?

Routing algorithms can be characterized in several different ways. One way is according to whether they attempt to obtain the optimal routes with respect to given objective functions or whether they use heuristics. Routing algorithms can also be characterized according to whether they are implemented in either a centralized or a decentralized manner. Some of the issues present in the design of routing algorithms are described below in the context of both of these characterizations.

Optimal routing algorithms are developed by setting up the problem as a nonlinear optimization problem where the objective is to minimize expected delay. The flow deviation algorithm is an example of an optimal centralized algorithm. It requires the availability of the first derivative of the expected delay with respect to the flows. [FRAT 73] and [CANT 74]. A decentralized optimal algorithm based on the same ideas was developed by Gallager. [GALL 77] Only one assumption, that the expected delays are convex functions of the flows, is made. The derivatives are estimated from measurements. As no one has implemented this algorithm, it is not clear whether it is practical or not. One problem is that the algorithm is iterative in nature and may require several iterations before it converges on the best routes. Consequently, it may not adapt to changes in the traffic very well. One nice property of the algorithm is that it allows for bifurcation in the routes. Instead of requiring all packets from one source to a destination to take exactly one route, it recognizes the advantages of spreading the traffic over several routes to equalize load.

The ARPANET routing algorithm is an example of a heuristic centralized algorithm. [MCQU 80] Here updates on link delays are determined over 20-second intervals and, if they deviate significantly from the previous estimate, are circulated around the network. All nodes store the entire topology along with the most recent link delays. Each node solves a shortest path problem using delays as weights to determine the routes. These routes are determined in one step after which they are fixed until the subsequent update cycle. As we understand the algorithm, no bifurcation is allowed. (Bifurcation could be allowed by considering approximate ties.) Decentralized variants of the ARPANET algorithm have been proposed by numerous authors. [MERL 79]

The previous discussion raises several issues. What kind of updates are necessary between gateways? Should they be global (i.e., all gateways get identical information) or not? What kind of information (i.e., estimates of link delays, estimates of first derivatives, estimates of the queueing model representations of various nodes) are required?

Two implementation approaches have been studied in the past, the decentralized (distributed) approach and the centralized approach. Under the first approach all gateways take part in executing the algorithm. Moreover, each gateway uses a portion of all the information required to execute the algorithm. This approach is typified by algorithms found in [GALL 77, MERL 79]. A centralized algorithm requires that all necessary information be collected at one processor. The algorithm is then executed at that processor using this information. The centralized approach can also be implemented whereby all information is collected at all gateways and each gateway executes the same algorithm. This second approach is currently used by ARPANET. [MCQU 80].

The decentralized approach produces algorithms that require several iterations before the optimal routes are produced. There is a question regarding how many iterations are required for convergence.[BERT 80] There is also a question regarding the length of time required to perform a single iteration. The answers to both of these questions will

determine how well decentralized algorithms can handle changes in workload and in topology.

As the decentralized approach exemplified by Gallager's algorithm may converge too slowly, it may be necessary to study heuristic solutions as exemplified by the ARPANET approach. The current ARPANET algorithm requires a single iteration. It does not, however, provide the routes that minimize expected delay. Consequently there is the question of how good the routes are that are supplied by such an algorithm.

The above discussion raise several additional issues that need to be addressed during the design of any implementable routing algorithm. Optimal algorithms should provide better routes in systems that do not change in time. However, they may not adapt well to changes in the system load or topology. Heuristic techniques, on the other hand may be able to handle such dynamic situations better. Which of these techniques is best suited to WIS? What kind of updates (if any) are required between gateways? There is also the issue regarding decentralized versus centralized algorithms. Conceivably, optimal algorithms will be better able to adapt to system changes because they will require a single iteration whereas decentralized algorithms typically require several iterations. On the other hand, centralized algorithms may require that more information be transported over the network than required by a decentralized algorithm. Furthermore, centralized algorithms may be more vulnerable to failures. The question arises regarding the tradeoff between these two approaches.

The above approaches have been directed primarily to networks with a single class of service. There is some preliminary work on extending the centralized approach to problems with constraints ([KUNG 83] and [JAFF 84]) and to problems with priorities. [HANT 85] The most important issue facing respondents to this request for proposal is that of developing routing algorithms for different classes of service and for priorities.

Much of the previous work has been concerned with developing efficient algorithms for networks with arbitrary topologies ([TAJI 77] and [GAFN 81]). This is not an issue in the case of the WIS system as the topology is very simple, as illustrated in Figure 6. This should facilitate the task of the designer and allow him to focus on some of the issues stated above. A good summary of work on routing can be found in [SCHW 80.] Two works that deal with non-bifurcated flows are [COUR 81] and [GAVI 83].

4.3 Description of the Algorithm Problem

4.3.1 Metrics

Several metrics should be considered in evaluating routes and performance. These include average delays, maximum allowable delays, probability of delay exceeding a value, throughput, and reliability (the probability that a packet will be delivered). Traffic will also be characterized by priorities. These could be fixed or related to grade of service. Some could require high reliability, others low delay. The priorities could change with time, e.g., the remaining delay. These metrics will appear as multiple objectives and or multiple constraints. The main concern of this effort is on how to choose routes, what information to pass between LAN's and DDN, how to operate the gateways to achieve performance, and protocols to tie all this together.

4.3.2 Topology

The topology of the network is very simple. There are no loops provided that DDN is loop free. All paths are from source LAN to source gateway to DDN to destination gateway to destination LAN. This is one hop (or three hops if you include LAN's as a hop).

Assumptions should be made initially that:

a. The DDN is large enough to accommodate WIS requirements.

b. That WIS traffic is a small part of DDN's total.

c. That WIS routing decisions do not affect DDN performance.

Thus, DDN performance metrics---delays and reliability---will not be affected by WIS routes. The DDN performance metrics will change however, and these changes will be reported to the gateways or determined by the gateways by probing.

Similarly it may be useful to ignore delays in the LAN's and concentrate on the gateways. Initially the gateways can be modeled as having two non-interfering halves---one for incoming and the other for outgoing messages. (Later these assumptions could be relaxed.) Thus, delays occur in three places, at the outgoing half of gateways, on the DDN, and in the incoming half of gateways. The performance of an outgoing gateway half is dependent only on routing decisions and traffic at that gateway. The performance of a DDN path is unaffected by WIS actions and must be reported to WIS gateways periodically. Performance of an incoming gateway is dependent on routing and traffic from all LAN's. This is the critical information that must be returned to source gateways for routing decisions. The study should be concerned with what information to return and how.

Most, if not all, distributed routing procedures were concerned with a vastly more complex topology. Much of their concerns were on how to avoid looping, how to propagate information over many hops, etc. Here the topology is essentially trivial. The designer should take advantage of this to concentrate on other issues - multiple objectives, constraints, priorities, iteration of algorithms, etc.

4.3.3 A Simple Model

To start, consider the network of Figure 6, with an average delay objective, no constraints or priorities, and only one class of traffic. Let there be n LAN's, each having two gateways to connect to DDN. Assume each gateway has only one connection to DDN. DDN has already selected a path between each gateway. There are several paths between each pair of LAN's. These depend upon source gateway, destination gateway, DDN path, and DDN access from source gateway if there is a choice. Let the delay on DDN of the kth path from LAN i to LAN j be $R_{[i,j,k]}$. This is a constant until a change is reported and is

not a function of WIS routing and traffic. Let the flow from LAN i to LAN j be $r[i,j]$ packets per second. The routing will divide this flow such that $r[i,j,k]$ uses path $p[i,j,k]$, where

[1]

$$\sum_{k} \left[r\,[i,j,k] \right] = r\,[i,j] \qquad \text{where } i,j = 1 \ldots n$$

Let $a[i,j,k]$ be the label of the source gateway used by path $p[i,j,k]$, here either 1 or 2. Similarly, let $b[i,j,k]$ denote the destination gateway. Let $s[i,l]$ be the total outgoing flow on gateway l of LAN i, i.e., gateway $g[i,l]$. Then

$$s\,[i,l] = \sum_{\substack{j,k: \\ a[i,j,k] = l}} \left[r\,[i,j,k] \right]$$

Let $t[j,m]$ be the total incoming flow in gateway $g[j,m]$.

$$\text{Then } t\,[j,m] = \sum_{\substack{i,k: \\ b[i,j,k] = m}} \left[r\,[i,j,k] \right]$$

Let $S[i,l] = S[i,l](s[i,l])$ be the outgoing delay on gateway $g[i,l]$. Let $T[j,m] = T[j,m](t[j,m])$ be the incoming delay on gateway $g[j,m]$. Then the total average network delay times the total network flow r is

[2]

$$D = \sum_{i,j,k} \left[r\,[i,j,k] * R\,[i,j,k] \right] + \sum_{i,l} \left[s\,[i,l] * S[i,l]\,(s\,[i,l]) \right] +$$

$$\sum_{j,m} \left[t\,[j,m] * T[j,m]\,(t\,[j,m]) \right]$$

$$\text{where } r = \sum_{i,j} \left[r\,[i,j] \right]$$

Let $C_{[i,l]}(s_{[i,l]}) = s_{[i,l]} * S_{[i,l]}(s_{[i,l]})$

and $D_{[j,m]}(t_{[j,m]}) = t_{[j,m]} * T_{[j,m]}(t_{[j,m]})$.

Then

[3]

$$D = \sum_{i,j,k} \left[r_{[i,j,k]} * R_{[i,j,k]} \right] + \sum_{i,l} \left[C_{[i,l]}(s_{[i,l]}) \right] + \sum_{j,m} \left[D_{[j,m]}(s_{[j,m]}) \right]$$

The problem is to choose the $r_{[i,j,k]}$, subject to [1], to minimize D. Adding [1] as constraints, using Lagrange multipliers, the following function should be minimized.

[3a]

$$J = D + \sum_{i,j} \left[q_{[i,j]} \left(r_{[i,j]} - \sum_{k} r_{[i,j,k]} \right) \right]$$

Taking the partial derivative with respect to $r_{[i,j,k]}$, yields the Kuhn-Tucker conditions,

[4]

$$R_{[i,j,k]} + \sum_{l=a_{[i,j,k]}} \left[C_{[i,l]}'(s_{[i,l]}) \right] + \sum_{m=b_{[i,j,k]}} \left[D_{[j,m]}' + (t_{[j,m]}) \right] >= q_{[i,j]}$$

For All k

The equality holds if $r_{[i,j,k]} > 0$. The ' denotes differentiation, e. g., C'(s) = dC(s)/ds. The first term in [4] is reported by DDN. The second term is obtainable at the source LAN. The third term is the key to distributed routing algorithms. In Gallager's procedure [GAL 77] The D'(t) would be sent periodically to source LAN's. Each source LAN would adjust its $[r_{[i,j,k]}]$ to try and satisfy [4]. For each [i,j], the $r_{[i,j,k]}$ for which the left hand side term of [4] is largest would be reduced and the others increased. Gallager proves convergence of a particular way of doing this.

There are many unanswered questions regarding this simple problem. Are there other distributed approaches for this simple topology? Is D'(t) the right quantity to send back? What are good iterative strategies for convergence of [4]? How much information needs to be sent back and how frequently should it be sent back? Will this overload the network?

4.3.4 Centralized Algorithms

If one knew the derivatives C'(s) and D'(t) for all s,t, all $R[_{i,j,k}]$, and all $r[_{i,j}]$ at one site, then [4] could be solved and the required $r[_{i,j,k}]$ found. A classic way of solving [4] is to use flow deviation ([FRA 73] and [CAN 74]). For any i,j, [4] says that the partial derivative of the total delay with respect to $r[_{i,j,k}]$ must be equal for all k for which $r[_{i,j,k}] > 0$. A feasible flow satisfies [1]. Start with a feasible flow and compute the left hand side of [4] for all i,j,k. Consider these as path lengths (costs). For each i,j find the path k with minimum length. Use this path for $r[_{i,j}]$. Let $f[_0]$ be the original feasible flow vector and $f[_1]$ be the flow just found. Let a new flow $f[_2] = u*f[_0] + (1-u)*f[_1]$. Find u , $0 <= u <= 1$, to minimize D and repeat.

Are there any sensible ways to perform this procedure? The $R[_{i,j,k}]$ and $r[_{i,j}]$ could be sent to all LAN's. This raises the question of whether the advantages of executing a centralized procedure outweighs the disadvantages of transmitting additional data over the network. How are $C'(_s)$ and $D'(_t)$ to be determined? We assume the gateways will be complex entities, especially with priorities and multiple classes. Can these derivatives be estimated or parameterized? It is interesting to contrast this approach with a distributed approach as described above. In a static environment they both converge to the same or equivalent solution. How do they behave when slow or rapid updates or changes have to be made? Are there approaches that combine the best features of these two, especially considering the simple topology? How do they extend to the complications of multiple constraints, objective functions, priorities, classes, and other objective functions?

4.3.5 Multiple Objectives

Consider a very simple example with one source and two possible paths. Each path can be modelled as an M/M/1 queue so that the delay on a path is 1/(c-r) where c is the capacity and r the flow. Let the probability of a packet being successfully sent on a path be P. Then if r is the total flow, $r[_i]$ is routed over path i, i =1,2, such that $r[_1] + r[_2] = r$, the average delay times flow is given by $D = r[_1]/(c-r[_1]) + r[_2]/(c-r[_2])$. The average probability of success P, or reliability, is given by $r*P = r[_1]*P[_1] + r[_2]*P[_2]$. Assume $P[_1] > P[_2]$. Minimizing delay alone gives $r[_1] = r[_2]$. Minimizing reliability alone gives $r[_1] = r$, $r[_2] = 0$. As a function of $r[_1]$, rP is linearly increasing from $r*P[_2]$ at $r[_1] = 0$ to $r*P[_1]$ at $r[_1] = 1$. rD has a unique minimum at $r[_1] = r/2$. Clearly $r[_1] < r/2$ is never optimum since both objective functions can be improved by using $r[_1] = r/2$. Any value of $r[_1]$ between r/2 and 1 is a non-dominated solution providing a trade-off between the two objectives. They are each minimized at opposite ends of the region. Without more specification, it is not possible to decide on a routing. One goal could be to find non-dominated regions as we just did. See [LAUE 79] and [BURC 83]. Another technique is to impose further constraints such as that the delay must be less than a certain value. What are suitable objectives and constraints for the WIS routing problem? How can non-dominated routings be found?

4.3.6 Constraints

This section considers the effects of adding constraints to the simple problem described in Section 4.3.3. Let $P[_{i,j,k}]$ be the probability of successful delivery of a packet by DDN on path $p[_{i,j,k}]$ and reported by DDN. The reliability for $r[_{i,j}]$ traffic, $p[_{i,j}]$, is given by

$$r_{[i,j]} * P_{[i,j]} = \sum_{k} \left[r_{[i,j,k]} * P_{[i,j,k]} \right]$$

The constraints that $r_{[i,j]}*P_{[i,j]} > w_{[i,j]}$ is to be added to the problem of Section 4.3.3. This is done by adding the additional multipliers to [3a]:

$$\sum_{i,j} \left[v_{[i,j]} \left(w_{[i,j]} - \sum_{k} \left[r_{[i,j,k]} * P_{[i,j,k]} \right] \right) \right]$$

This adds to [4] the term $-v_{[i,j]}*P_{[i,j,k]}$ for each i,j,k. The same iterations for distributed and centralized algorithms can be used but now there is a bias $v_{[i,j]}*P_{[i,j,k]}$ in each term. The $v_{[i,j]}$ are not known. They are selected to satisfy the constraints. Another iteration can be proposed. Start with a set of $v_{[i,j]}$'s. Solve for a set of flows and check the constraints. If they are satisfied reduce the appropriate $v_{[i,j]}$ and repeat. If they are not satisfied increase $v_{[i,j]}$ and repeat. There are centralized and distributed versions of this procedure. In the centralized version sufficient information must be passed to evaluate the constraints.

Multiple constraints in centralized routing have been discussed by [KUNG 83] and [JAFF 84]. It is also a standard problem in optimization. What techniques apply to our problem? Does the iteration discussed above converge? Is it feasible?

4.3.7 Priorities

Some priorities can be handled in the above framework. The flows, delay functions, and delays, etc., are now indexed by priority as well. A complication is that the objective functions may no longer be convex. [HANT 85] Thus, finding optimal solutions becomes a very hard problem. As an example, consider two priorities and the delay of each as multiple objectives. The delay of the higher priority is dependent upon the lower priority traffic unless preemption is assumed. The delay of the lower priority also depends upon higher priority traffic. The delays are not convex functions of the flow. Thus, the problem has multiple objectives and a more difficult objective function to optimize. Dealing with priorities raises many of the same questions as were encountered with multiple objectives and constraints.

4.4 Overall Objectives

There are several aspects of inter-LAN communications that need to be addressed before an inter-LAN mechanism can be implemented. First, a good routing protocol must be developed. Second, the gateway must be designed for high performance. The routing protocol itself consists of two components: an algorithm for computing the routes based on some cost function, and the protocol that uses this route information to perform the inter-LAN communications. A multiple phase effort on inter-LAN communications is necessary in order to achieve the integration of these routing algorithms and protocols into WIS. Table I outlines the efforts that should be undertaken during these phases.

Table I. Development Strategy

PHASE	ROUTING		
	ALGORITHM	PROTOCOL	GATEWAY
1	DEVELOP AND EVALUATE PROTOTYPE ALGORITHMS BY CONTRACTOR	ISSUES DEFINED BY COTR	
2		SEPARATE CONTRACT STUDY	
3	INTEGRATION INTO WIS		

The algorithms and protocols developed during the first two phases would be developed, integrated into WIS, and evaluated on a testbed system using the WIS standard Ada PDL with ANNA extension.

4.5 Conclusion

The objective of this project is to develop efficient routing algorithms that will provide good routes within WIS under varying traffic loads and topological changes. The algorithms will be implemented and evaluated for performance and efficiency by running them on typical problems. These problems are to be described in the proposal. During the initial phase the details of which problems to consider will be resolved.

The routing algorithms should be concerned with multiple objective functions, constraints, and priorities. Different classes will require different services and will be routed according to criteria associated with each class. The routing algorithms will have to accommodate a number of different classes. The designer will evaluate algorithms with respect to optimality, performance, speed of convergence, amount and frequency of information transfer, computational and storage requirements, and interaction with other elements of WIS.

Distribution List for IDA Paper P-1893

Sponsor

Maj. Terry Courtwright — 5 copies
WIS Joint Program Management Office
7798 Old Springfield Road
McLean, VA 22102

Maj. Sue Swift — 5 copies
Room 3E187
The Pentagon
Washington, D.C. 20301-3040

Other

Col. Joe Greene — 1 copy
STARS Joint Program Office
1211 Fern St., Room C107
Arlington, VA 22202

Defense Technical Information Center — 2 copies
Cameron Station
Alexandria, VA 22314

CSED Review Panel

Dr. Dan Alpert, Director — 1 copy
Center for Advanced Study
University of Illinois
912 W. Illinois Street
Urbana, Illinois 61801

Dr. Barry W. Boehm — 1 copy
TRW Defense Systems Group
MS 2-2304
One Space Park
Redondo Beach, CA 90278

Dr. Ruth Davis — 1 copy
The Pymatuning Group, Inc.
2000 N. 15th Street, Suite 707
Arlington, VA 22201

Dr. Larry E. Druffel — 1 copy
Software Engineering Institute
Shadyside Place
580 South Aiken Ave.
Pittsburgh, PA 15231

Dr. C.E. Hutchinson, Dean — 1 copy
Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

Mr. A.J. Jordano — 1 copy
Manager, Systems & Software
Engineering Headquarters
Federal Systems Division
6600 Rockledge Dr.
Bethesda, MD 20817

Mr. Robert K. Lehto — 1 copy
Mainstay
302 Mill St.
Occoquan, VA 22125

Mr. Oliver Selfridge — 1 copy
45 Percy Road
Lexington, MA 02173

IDA

General W.Y. Smith, HQ — 1 copy
Mr. Seymour Deitchman, HQ — 1 copy
Mr. Robin Pirie, HQ — 1 copy
Ms. Karen H. Weber, HQ — 1 copy
Dr. Jack Kramer, CSED — 1 copy
Dr. Robert I. Winner, CSED — 1 copy
Dr. John Salasin, CSED — 1 copy
Mr. Mike Bloom, CSED — 1 copy
Ms. Deborah Heystek, CSED — 1 copy
Mr. Michael Kappel, CSED — 1 copy
Mr. Clyde Roby, CSED — 1 copy
Mr. Bill Brykczynski, CSED — 1 copy
Ms. Katydean Price, CSED — 2 copies
IDA Control & Distribution Vault — 3 copies

END

4-87

DTIC